IEEE MICRO

AUGUST 1991

Chips, Systems, Software, and Applications

Far East Issue

Special Feature

An Expert System for Integrated Hardware/Software Design

• Fuzzy Logic • Curbing Chip Piracy

1951-1991
40 YEARS OF SERVICE
IEEE COMPUTER SOCIETY

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

IEEE MICRO

Published by the IEEE Computer Society

August 1991

FEATURES

Circulation: *IEEE Micro* (ISSN 0272-1732) is published bimonthly by the IEEE Computer Society, PO Box 3014, Los Alamitos, CA 90720-1264; IEEE Computer Society Headquarters, 1730 Massachusetts Ave., NW, Washington, DC 20036-1903; IEEE Headquarters, 345 East 47th St., New York, NY 10017. Annual subscription: $21 in addition to IEEE Computer Society or any other IEEE society member dues; $38 for members of other technical organizations. This journal is also available in microfiche form.
Postmaster: Send address changes and undelivered copies to *IEEE Micro*, PO Box 3014, Los Alamitos, CA 90720-1264. Second-class postage is paid at New York, NY, and at additional mailing offices.

DEPARTMENTS

Reader Interest/Service/ Subscription cards, p. 64A; Call for papers, p. 93; CS membership application, p. 95; Advertiser/ Product Index, p. 96; Computer Society information, cover 3

IEEE Computer Society
PO Box 3014, Los Alamitos, CA 90720-1264
(714) 821–8380

Editorial: Unless otherwise stated, bylined articles and descriptions of products and services reflect the author's or firm's opinion; inclusion in this publication does not necessarily constitute endorsement by the IEEE or the IEEE Computer Society. Send editorial correspondence to *IEEE Micro*, PO Box 3014, Los Alamitos, CA 90720-1264. All submissions are subject to editing for style, clarity, and space considerations.

* Submit six copies of all articles and special-issue proposals to Dante Del Corso, Dipartimento di Elettronica, Politecnico di Torino, C.so Duca degli Abruzzi, 24, 10129 Torino, Italy; phone +39-11-556-4044; Compmail: d.delcorso; Internet: delcorso@polito.it; Bitnet: delcorso@itopoli
or
** Ashis Khan, Mips Computer Systems, Inc., 950 DeGuigne Drive, Sunnyvale, CA 94086; Internet: ashis@mips.com.

From the Editor-in-Chief

Congratulations!

THIS IS AGAIN A THEME issue, so I leave the technical presentation to Guest Editor Ken Sakamura. Do not miss his introduction; he has long experience in collecting the most significant developments in the microcomputer and microelectronics areas in Japan. These advances are worth watching carefully.

You will find news of the recent change in the editorial board in another part of the magazine; what I'd like to bring to your attention is that the board has assigned the Best Article Award for 1990. The choice is "The Motorola 68040 Processor" by R.W. Edenfield, M.G. Gallup, W.B. Ledbetter Jr., R.C. McGarity, E.A. Quintana, and R.A. Reininger. This article appeared in two parts in the February and June 1990 Hot Chips issues of *Micro* with Guest Editor Robert Stewart. Hearty congratulations to all involved!

This article was also chosen on the basis of your comments on the Reader Service Cards; you *can* influence the selection, so please continue to offer feedback, other comments, and proposals. We welcome them.

Turn to p. 93 for the Mailbag.

Futurebus+ correction

The article, "Real-Time Computing with IEEE Futurebus+" by L. Sha, R. Rajkumar, and J.P. Lehoczky published in the June issue, carried one error so big that it went through proofreading without being noticed: The sentence "... indicates that Futurebus+ can realize 100,000 transfers of ... per second." (page 30, first column, row 17) should read: "... indicates that Futurebus+ can realize 100,000,000 transfers of ... per second." Just three zeroes make a big difference.

We apologize to the readers, the authors (the original manuscript was correct), and the Futurebus+ committee. That 100,000-transfer-per-second figure was better suited to first-generation microcomputers than "Future-generation" ones!

0272-1732/91/0800-0002$01.00 © 1991 IEEE

The first chip-layout copying case

Richard H. Stern

Law Offices of

Richard H. Stern

1300 19th Street NW,

Suite 400

Washington, DC 20036

Just about a year ago, a San Diego federal jury returned a verdict of more than $25 million for the plaintiff in the first chip-layout copying case decided so far—*Brooktree Corp.* v. *Advanced Micro Devices, Inc.* Congress passed the Semiconductor Chip Protection Act (SCPA) in 1984 to stop chip piracy. The law must have been effective in curbing chip piracy, because only two cases have been brought so far raising any substantial SCPA issue—Intel's suit against AMD over the 80287 coprocessor in 1990[1] and the Brooktree-AMD case.

In the Brooktree case, AMD asked the court either to grant a judgment in its favor, notwithstanding the jury's verdict, or to grant it a new trial. AMD's position was that no reasonable jury could have decided the case against AMD on the factual record made at trial. The judge denied those motions in December 1990 and left the jury's verdict intact. AMD then appealed the case to the US Court of Appeals for the Federal Circuit. That court will hear oral arguments this fall and will probably decide the case by early 1992.

Since this is the first judgment rendered under the new chip law, the trial court had to make some first decisions about what the new law means. AMD is challenging some of those decisions in its appeal. The appellate court will therefore provide the semiconductor industry with the first authoritative judicial interpretation of SPCA's meaning.

The dispute between Brooktree and AMD started in 1988 when AMD introduced its Am81C458 RAM/digital-analog converter (RAMDAC) chip as a lower priced substitute for Brooktree's Bt458 color-palette video chip. AMD claimed that it had reverse-engineered the design, but Brooktree cried "piracy."

Brooktree sued and then tried to get the court to issue a preliminary injunction against AMD, which would have stopped AMD from selling its Am81C458 chips while the suit was pending. But the court found that Brooktree failed to show that it was sufficiently likely to prevail on the merits to justify the preliminary injunction. The court therefore allowed AMD to continue to sell its chips while the case worked its way through the judicial system. That 1988 result was reversed, however, after the jury found in Brooktree's favor in 1990.

One-cell controversy

A particularly interesting aspect of the controversy is that it is limited to only one cell of the RAMDAC chip—in fact, to two layers of that cell. According to Brooktree, the key element of the color-palette chip is its multiport static RAM (SRAM) cell, which has a layout that permits operation at the previously unheard-of speed of 125 MHz. Brooktree claims that AMD copied two of the nine layers of this cell, the active and polysilicon layers, and that this copying violated the SCPA.

Figures 1 to 4 show the disputed cell layers. Figure 1 shows the active layer of several SRAM cells within the Brooktree Bt458 chip, with a box surrounding one cell. Figure 2 is a similar depiction of AMD's Am81C458. Figure 3 shows the polysilicon layer of several SRAM cells within the Brooktree Bt458 chip, with a box around one cell. Figure 4 is a similar depiction of AMD's Am81C458.

At the trial, AMD claimed that the SCPA does not protect cells, and that it only protects entire chip layouts. That position is untenable and the court rejected it. Presumably, AMD will not even bother to appeal that issue, because the legislative history of the SCPA contains a number of

0272-1732/91/0800-0003$01.00 © 1991 IEEE

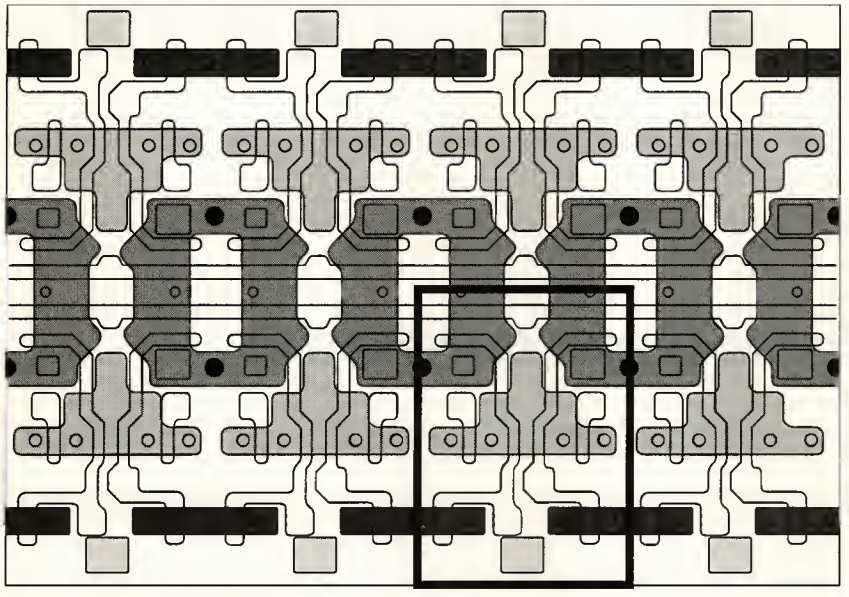


Figure 1. Active layer of the Brooktree Bt458 chip.

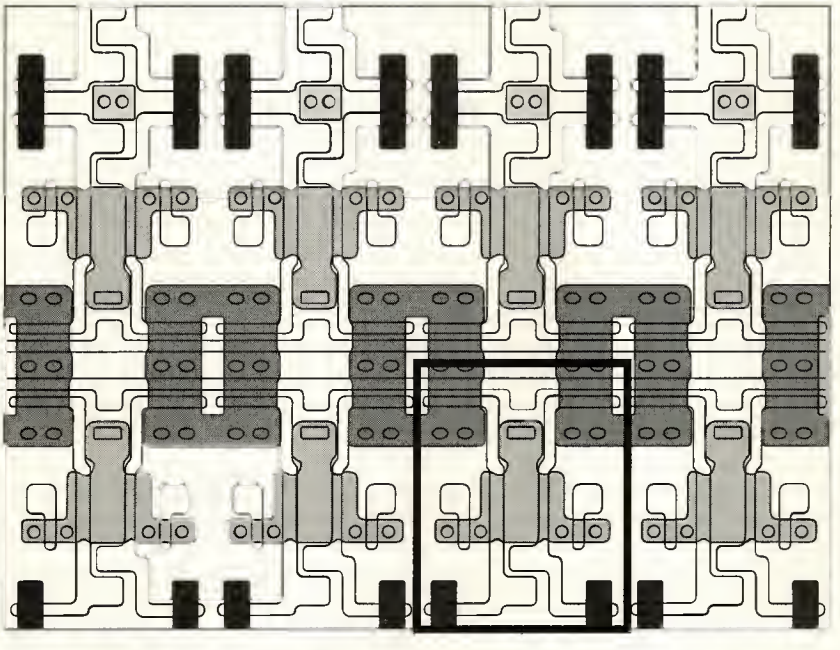


Figure 2. Active layer of the AMD Am81C458 chip.

statements that say the law intends to protect cells as well as whole chips. In this case, both companies replicated the 10-transistor SRAM cell about 6,000 times on the chips, accounting for about 30 percent of the surface area and 80 percent of the total transistor count in the cell. More important, Brooktree claimed that the SRAM cells were "the heart and soul of the chip." What "heart and soul" meant was not clear, but it certainly must have sounded impressive to the jury.

The issue of cell layers is more controversial. Nothing in the SCPA or its legislative history says that Congress wanted to protect layers of chips, as contrasted with whole chips, or to protect layers of cells. The committee reports speak of protecting material, or important, portions of a chip. But then the reports refer to counter and oscillator modules, and to cells from cell libraries (that is, to vertically partitioned portions of chips). They contain no statements about protecting portions of modules or cells (horizontally partitioned portions of chips). Therefore, it is open to AMD to contend on appeal that the jury had no valid basis to find against AMD in this case, because the evidence of similarity in the SRAM layouts was too sketchy.

Both parties dealt with the similarity issue by calling on the testimony of experts, each of whom the court said was credible. On some points, only one expert said anything. On others, the parties' experts contradicted one another (albeit "credibly"). The net legal effect was that the jury was free to choose between the factual points to which the parties' experts differed, but where only one expert discussed a point the jury was not free to find the opposite of what he said. To appreciate how that works, consider a case in which the plaintiff provides evidence to prove factual points A and B, and the defendant puts in contradictory evidence on point A, but nothing on B. The jury can then find either way on fact A, but it must find for the plaintiff on fact B because the only evidence before the jury on that point was the plantiff's uncontroverted evidence. Uncontroverted evidence must be credited unless there are special circumstances.

Brooktree's evidence centered on the two layers pictured in Figures 1 to 4. Brooktree did not produce testimony that the other seven SRAM layers were

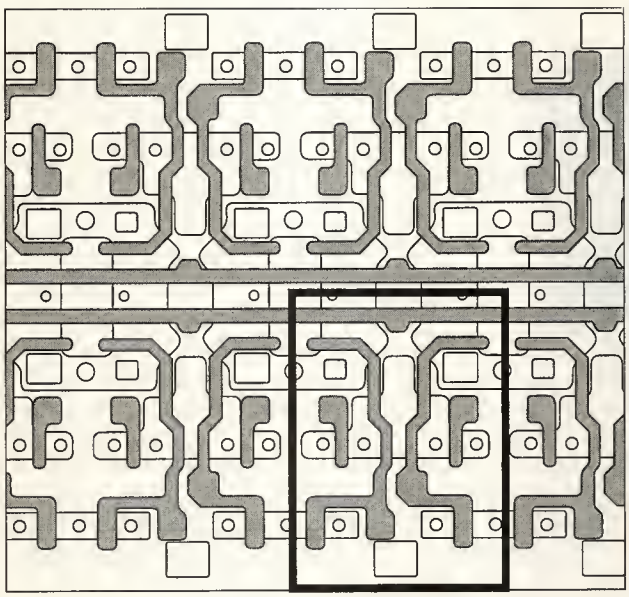


Figure 3. Polysilicon layer of the Brooktree Bt458 chip.

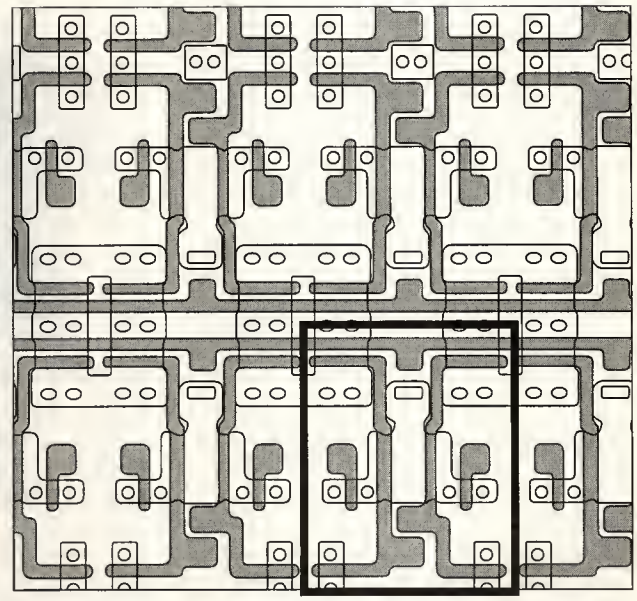


Figure 4. Polysilicon layer of AMD Am81C458 chip.

similar in layout. Brooktree's expert said that the Brooktree and AMD SRAM cells were "substantially similar" because the transistors were grouped together in the same way. He admitted, however, that there were differences. For example, one part of Brooktree's cell (Figure 1) has a 45-degree outpost jutting from its sides while AMD's cell (Figure 2) has straight sides. Brooktree's expert also conceded that additional differences existed in other layers, such as the source drain diffusion layers (not shown here). It does not appear that Brooktree presented any testimony that the cells of Figures 1-4 were not only "substantially similar" but also "substantially identical."

AMD's expert testified that the cells of Figures 1-4 were not substantially identical and were not even substantially similar. The testimony about lack of substantial similarity is immaterial, presumably, since it appears that the jury must have believed the testimony of Brooktree's expert that the cells were substantially similar. The testimony about lack of substantial identity, however, appears to be uncontroverted, since Brooktree's expert seems not to have addressed the "substantially identical" issue at all. (I have not made a detailed enough study of the trial evidence to know whose version of "who struck John" to believe. I base my comments, therefore, on the limited information that was readily available, which compels frequent use of "seems" and "probably.")

AMD's expert pointed to differences in orientation of P-channel transistors (horizontal bands in Figure 1 versus vertical bands in Figure 2), different transistor sizes, an extra metal ground line in AMD's SRAM, and different P-well contacts. AMD's expert not only denied substantial similarity of the active and polysilicon layers but also denied substantial similarity in the P+ implant and other layers (on which Brooktree's expert apparently said nothing).

In addition to the testimony of the rival experts, the trial evidence included actual plots of the other layers. In theory, the jury might have decided whether the chips' other layers were substantially identical on its own. But that seems to make no sense. How can anyone but an expert understand what is a material difference and what is a trivial difference in an SRAM layout? Without expert testimony on the layout, a jury (or judge, for that matter) is really helpless, except for the most clear cases. The SCPA's legislative history acknowledges that fact several times.

As I interpret the significance of the foregoing evidence, the upshot is that the record made at trial before the jury probably permitted the jury to make three findings.

First, the active and polysilicon layers were substantially similar, and presumably the jury so found. To the extent that the verdict relies on that finding, the court did not make a reversible error in letting the jury rely on it. The record would support a verdict based on such a finding.

Second, AMD's expert provided the only evidence on whether those layers were not only substantially similar but also substantially identical, and he denied it. Therefore, the jury was not entitled to find substantial identity in the active and polysilicon layers. To the extent, if any, that the verdict must rely on a jury finding of substantial

identity, the verdict is not supported.

Finally, AMD's expert also presented the only comprehensible evidence on the other seven layers (P+ implant and so on), and he denied even substantial similarity. Therefore, the jury was entitled only to find a lack of substantial similarity in the other seven layers. To the extent, if any, that the verdict must be based on a finding that the other seven layers were substantially similar, the verdict is not supported by the evidence.

Where does all of that get us?

On the basis of these points, the appellate court may find two issues are important.

First, the plaintiff in this kind of case must prove, at a minimum, substantial similarity of a relevant part of a chip layout. Otherwise, the case should never go to the jury. In fact, the defendant should not even be obliged to put on a defensive case. The case should be dismissed at the close of the plaintiff's proof, because that proof is inadequate to justify a verdict for the plaintiff. Thus, if the legally relevant part of a chip layout must be, at least, all layers of a cell and cannot just be two out of nine layers of a cell, Brooktree's proof may not have entitled Brooktree to get to the jury. On the other hand, maybe two layers out of nine are enough, for example, for the jury to conclude that the layers were of major importance (remember that "heart and soul of the chip" business?). In that case, maybe Brooktree proved enough substantial similarity to get to the jury or at least require AMD to provide a defense.

I am not going to try to give you a definitive answer to the question, "Which standard about layers is right under the SCPA?" Resolving such issues is what makes for appeals and horse races. Instead, you might ponder whether a bacon, lettuce, tomato, and mayonnaise on rye sandwich is substantially identical, substantially similar, or neither, to an anchovy,

salami, tomato, and mayo on rye sandwich. The question is not wholly free from doubt. The better view seems to be that the SCPA does not contemplate basing infringement on a partial selection of strata—as contrasted with all strata—of a given cell or module. Nonetheless, a serious argument can be made on the other side of the issue.

A second issue that the appellate court may consider important is whether the two SRAM cells were similar enough to meet the requirements of a case involving reverse engineering. Say that the SCPA's legislative history tells us (and the court so stated in its opinion denying a preliminary injunction in this case) that the threshold evidentiary criteria for a case of reverse engineering are met. Then the legal test for infringement depends on whether the layouts are "substantially identical." That test calls for a closer similarity than the ordinary "substantial similarity" test involves. It means not just "a lot of similarity," but similarity in all but immaterial or trivial respects.

Therefore, *if* AMD met the threshold requirements for a case of reverse engineering, it probably should have prevailed—contrary to the San Diego jury's verdict for the hometown team. The italicized "if" in the first sentence is a very big "if." Whether AMD met the threshold and other requirements for the defense of reverse engineering is the next major issue in this case.

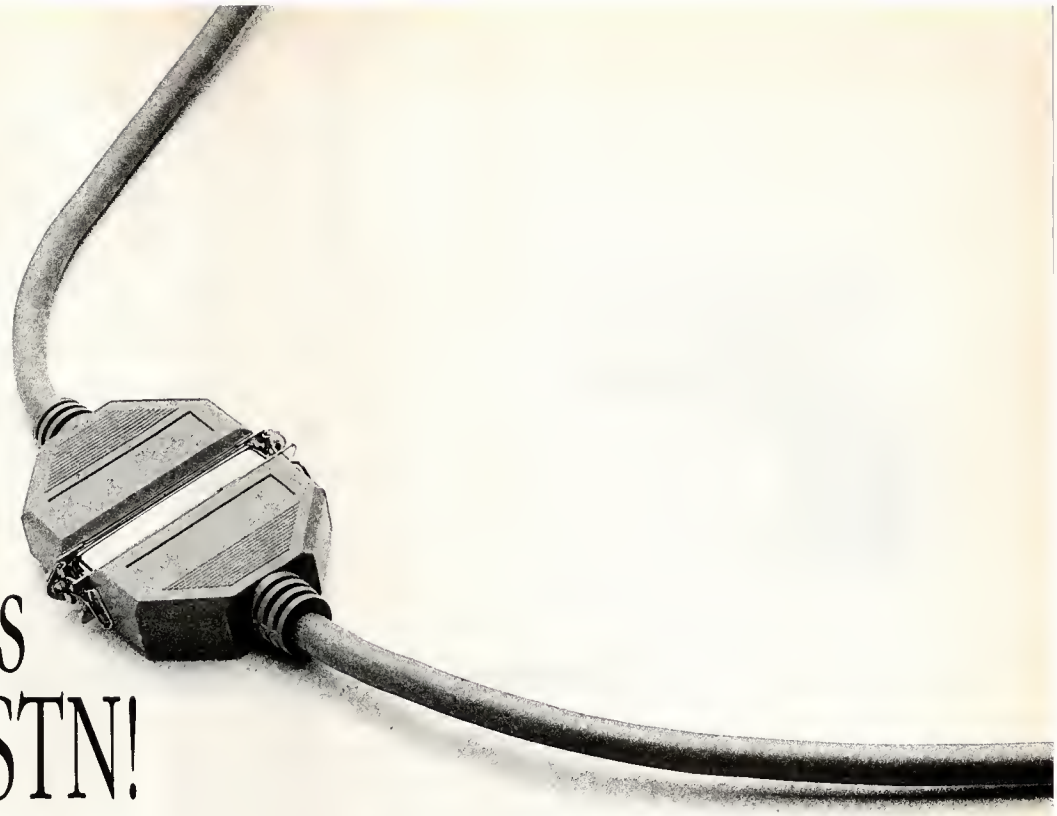
Sweat of the brow

The first and basic requirement for the defense of reverse engineering to apply in a case of infringement of mask work rights is that the accused infringer expended a great deal of time, effort, and money in studying and redesigning the chip. This is a "sweat of the brow" test. The SCPA wants to make second comers expend their own sweat of the brow, preferably (although not necessarily) developing enhancements in the course of their efforts. The SCPA, at the very least, wants to keep second comers from blatantly purloining (or plagiarizing) the fruits of the first comer's efforts. Typically, the best evidence of sweat of the brow is a paper trail of records showing (nonmythical) man-months spent, payroll, computer simulations, and other types of work documentation constituting the alleged reverse engineering.

In this case, AMD showed a big stack of paper. Brooktree challenged the stack's significance, however, on the ground that AMD spent most of its effort in chasing down a blind alley (a fewer-than-10-transistor version of the SRAM cell, which did not work). The court suggested, in denying AMD's motion for a new trial and judgment notwithstanding the verdict, that the jury may have considered that AMD's reverse-engineering efforts were not the kind that count under the SCPA. That is a possibility. Maybe the jury thought that time and effort spent on a "snipe hunt" does not count. If that view prevails on appeal, it would not matter that Brooktree failed to supply evidence of substantial identity, for that legal test becomes relevant only after the defendant passes the threshold test.

On the other hand, that may be a wrong view as a matter of law. The court assumed that this was a factual question to be decided by the jury. It is unclear, however, whether the SCPA's concept of reverse engineering allows otherwise qualifying time, effort, and money (sweat of the brow) to be disregarded simply because it is based on an erroneous design concept. It may be that as a matter of law, mak

continued on p. 94



Make Great Connections Online On STN!

On STN International® yau can access databases cavering every area af engineering. Yau'll find bibliagraphic and numeric files praduced by leading scientific arganizatians, like AIChE, Engineering Infarmatian, IEEE, Natianal Institute af Standards and Technalagy, and many athers.

And everything abaut STN has been created ta assist yau in abtaining this informatian efficiently and ecanamically. On STN, yau'll find special features which enhance yaur searching, whether yau're a navice ar an expert.

One Command Language — Using a few simple cammands, yau'll be able ta abtain infarmatian in mare than 100 databases an STN. And STN's saftware, Messenger,® enables yau ta carry a search created in ane database aver ta anather an STN far further infarmatian.

Element Term Index — Thraugh STN's Element Term (ET) Field, yau can search far chemical symbals and ather specialized natatians. Using the ET Field can increase yaur accuracy AND efficiency.

Numeric Searching — On STN, yau can search numeric values ta lacate substances having the praperty values yau specify; search numeric ranges ta find data yau might atherwise miss; chaase fram SI, metric, engineering, ar ather units ta display praperty values; search with substance names, names af praperties, ar CAS Registry Numbers® ta retrieve numeric data.

As an STN custamer, yau can receive help fram warkshops, tutarial diskettes, STN Express® saftware, tall-free Help Desk, newsletters, and anline dacument ardering. Na ane supparts yau like STN!

Make great cannectians an STN by filling out and returning the caupan belaw.

☐ **YES! Please tell me how to become a user of STN International.**

Name

Title

Organizatian

Address

City, State ZIP

Phone

MAIL TO: STN Internatianal, c/a Chemical Abstracts Service, Marketing Dept. 30091, P.O. Bax 3012, Calumbus, OH 43210
FAX TO: 1-614-447-3713

STN®
INTERNATIONAL
The Scientific & Technical Information Network

Advances in fuzzy theory and applications

David K. Kahaner

US Office of Naval

Research, Far East

kahaner@cs.titech.ac.jp

[George J. Klir of the State University of New York at Binghamton visited Japan to assess that country's involvement in fuzzy logic research. He offers the following comments for publication in this issue. I thank him for his generous, interesting contribution.—D.K.K.]

Fuzzy theory emerged in the second half of this century by challenging basic assumptions of three theories: sharp boundaries in classical set theory, classical (Aristotelian) logic that each proposition must either be true or false, and additivity in classical measure theory, particularly probability theory. (See the historical overview.)

By the 1980s fuzzy theory had gradually advanced due to different factors. 1) The theory sufficiently matured. 2) Several organizations promoting the theory and applications emerged (the North American Fuzzy Information Processing Society, International Fuzzy Systems Association, and Japan Society for Fuzzy Theory and Systems). 3) The theory became recognized as a respectable academic subject by a growing number of academic programs, which initiated and contributed to the publication of the first textbooks. 4) Above all, some applications of fuzzy theory became sufficiently appealing to attract the attention of industries and other nonacademic constituencies.

In its initial stage, however, fuzzy theory encountered a lot of skepticism and, on some occasions, open hostility. In the US some highly influential scholars became quite hostile during the early stages of its development. In contrast, some influential and highly respected Japanese scholars supported fuzzy theory, which in turn led to successful applications and further advances in the theory itself. As a result, Japan today is far ahead not only in the theory but above all in its practical utilization.

Current research

The motivation to support today's research in fuzzy theory and applications in Japan likely comes from some highly successful industrial applications of fuzzy control in the late 1980s. The first significant application appeared in the automatic-drive fuzzy control system for subway trains in Sendai City. Now generally praised as superior to other comparable systems based on classical control, the fuzzy controller achieves high precision in stopping at any designated point (to within 7 cm).

The great utility of fuzzy control can be seen in projects such as:

- chlorine controllers for water-purification plants,
- elevator control systems,
- traffic control systems,
- bulldozer control,
- air-conditioning systems,
- control systems for cement kilns, and
- control of working machines, vacuum cleaners, video cameras, refrigerators, and so on.

[Two recent fuzzy products include television meeting systems from Mitsubishi Electric Corporation and fuzzy and neuroelectric fans from Sanyo Electric, or Fisher as it is known outside Japan.

The fuzzy logic in Mitsubishi's product decides the degree of image movement, which is used in the decision about codec variables. Apparently,

0272-1732/91/0800-0008$01.00 © 1991 IEEE

An historical overview

The first challenge to classical logic theories came in the 1960s with the fuzzy set theory founded by Zadeh,[1] even though Black[2] envisioned some key ideas of the theory in 1937. Next came fuzzy logic, which emerged as an outgrowth of fuzzy set theory[3] and a generalization of the Lukasiewicz infinite-valued logic defined on the unit interval.[4] A third challenge appeared with fuzzy measure theory founded by Sugeno[5] in 1974, even though the basic ideas of fuzzy measures, monotonicity, and continuity were already present in Choquet capacities introduced in 1953.[6]

While early fuzzy theory encountered a lot of skepticism and, on some occasions, open hostility, it became quite strong in the 1970s.[7] New important concepts were introduced such as fuzzy numbers, fuzzy topology, and various kinds of fuzzy relations. An extension principle appeared in 1975[8] by which other concepts and theories of classical mathematics can readily be "fuzzified." Researchers developed a theory of dynamic fuzzy systems, investigated operators for aggregating fuzzy sets in a comprehensive way, introduced fuzzy sets of more general types, and formulated various categories of fuzzy sets and relations in category-theoretic terms.

All these advances influenced the development of fuzzy logic. For example, fuzzy arithmetic is crucial for dealing with fuzzy quantifiers, and fuzzy operations of complementation, union, and intersection can readily be mapped into the corresponding logic operations of negation, disjunction, and conjunction. Also, fuzzy relation equations play an important role in implementing fuzzy rules of inference.

In particular, several theories that generalize or complement probability theory were introduced, including fuzzy events,[9] random set theory,[10] Sugeno theory of measures,[5] Dempster-Shafer theory of evidence,[11] and possibility theory.[12]

Some ideas of prospective applications of fuzzy theory also emerged in the 1970s: fuzzy control,[13] fuzzy decision making,[14] and fuzzy pattern recognition.[15] These ideas were still of interest almost exclusively to the academic community. Industries, business, and government showed little interest in this new area. In spite of a general lack of interest, fuzzy theory continued to advance rapidly. Applications of the theory, however, remained hopelessly behind the theory itself until the 1980s.

References

1. L.A. Zadeh, "Fuzzy Sets," *Information and Control,* Vol. 8, No. 3, 1965, pp. 338-353.
2. M. Black, "Vagueness: An Exercise in Logical Analysis," *Philosophy of Science,* Vol. 4, 1937, pp. 427-455.
3. G.J. Klir and T.A. Folger, *Fuzzy Sets, Uncertainty, and Information,* Prentice Hall, Englewood Cliffs, N.J., 1988.
4. N. Rescher, *Many-Valued Logic,* McGraw-Hill, New York, 1969.
5. M. Sugeno, "Theory of Fuzzy Integrals and Its Applications, "PhD thesis, Tokyo Institute of Technology, 1974.
6. G. Choquet, "Theory of Capacities," *Annales de L'Insitut Fourier,* Vol. 5, 1953-54, pp. 131-295.
7. L.A. Zadeh, "Birth and Evaluation of Fuzzy Logic: Expectation of Japan's Role," *J. of Japan Soc. for Fuzzy Theory and Systems,* Vol. 2, No. 2, 1989, pp. 182-200.
8. L.A. Zadeh, "The Concept of a Linguistic Variable and Its Application to Approximate Reasoning," *Information Sciences,* Vol. 8, 1975, pp. 199-249 and 301-357; Vol. 9, pp. 43-80.
9. L.A. Zadeh, "Probability Measures of Fuzzy Events," *J. Math. Analysis and Applications,* Vol. 23, 1968, pp. 421-427.
10. G. Matheron, *Random Sets and Integral Geometry,* John Wiley and Sons, New York, 1975.
11. G. Shafer, *A Mathematical Theory of Evidence,* Princeton Univ. Press, Princeton, N.J., 1976.
12. D. Dubois and H. Prade, *Possibility Theory,* Plenum Press, New York, 1988.
13. M. Sugeno (ed.), *Industrial Applications of Fuzzy Control,* North-Holland, New York, 1985.
14. H.J. Zimmermann, *Fuzzy Set, Decision Making and Expert Systems,* Kluwer, Boston, 1987.
15. J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms,* Plenum Press, New York, 1981.

the fuzzy logic chooses the best values of 10 parameters in deciding image quality according to the image movement.

Though the concrete method of fuzzy logic implementation may be different, the basic philosophy seems to be quite similar to the movement-detection system in a Panasonic cam-corder. People at one site can hold a meeting with up to eight sites bidirectionally, and up to 64 sites can receive the meeting image at the same time unidirectionally. The price is 7.3-19.5 million yen.

Sanyo's fan finds the direction of its remote infrared master. Fuzzy reasoning infers the infrared strength of the remote controller, and neural networks decide the angle using three sensors and the estimated strength of infrared. Thus, the fan finds the position of the

user and sends cool wind in that direction.—D.K.K.]

Current activities

Following is an overview of current activities at several key organizations involved in research on fuzzy theory and applications in Japan.

LIFE. The Laboratory for International Fuzzy Engineering Research founded in 1989 and located in Yokohama is headed by Toshiro Terano, professor at Hosei University. Terano is one of the earliest and most important contributors to fuzzy theory in Japan.

LIFE comprehensively studies the many issues associated with the human friendliness of machines, especially two principal requirements: intelligence and smooth communication capabilities. LIFE's Decision Support Group investigates intelligent support systems for dealing with various problems involving large-scale systems models. As a concrete theme, the group chose to develop a prototype of a foreign exchange support system by which macroscopic predictions of exchange rates can be made on the basis of market participants and the economic situation.

The laboratory's Intelligent Robot Group seeks to develop a robot that combines sophisticated visual perception capability with the capability of understanding natural language. Fuzzy logic plays an essential role is this extremely challenging project. A second project develops high-level visual perception capabilities. The research centers on the knowledge-based model description of objects and appropriate reasoning methods to deal with the model and actual image data.

The Fuzzy Computing Group investigates various aspects of computer systems from the standpoint of fuzzy theory. One project focuses on fuzzy neural networks, aiming to combine advantages of fuzzy logic and neural network technology. This idea, which originated in the US with Bart Kosko, is being far more rapidly developed here than in the US. Additional projects include a fuzzy expert shell for building fuzzy expert systems and the development of a computer specifically designed for fuzzy information processing. The last project should result in specifications for the entire architecture of the computer and the necessary hardware and software technology to implement the architecture.

FLSI. Takeshi Yamakawa, professor of Kyushu Institute of Technology, directs the Fuzzy Logic Systems Institute, located in Iizuka and established in 1990. Yamakawa is well known for his pioneering work on hardware technology implementing fuzzy inference rules for fuzzy controllers. FLSI researches and develops high-speed hardware to support fuzzy logic. This natural outgrowth of Yamakawa's previous work on a 10-MFIPS (fuzzy inferences per second) electronic circuit seeks to design and build a fuzzy computer (in cooperation with LIFE). This computer would be capable of using rules of fuzzy logic at very high speed—Japan's sixth-generation computer.

FLSI develops high-speed hardware to support fuzzy logic.

Other research on fuzzy neural networks involves relevant theory, hardware development, and various applications. Currently, the main focus seems to be on hardware development of an artificial fuzzy neuron, a neuron in which fuzzy numbers rather than ordinary numbers represent the weights of synaptic junctions. Applications include pattern recognition of handwritten characters. One layer of fuzzy neurons, with each neuron designed for recognizing one particular character, implemented a hardware system for such recognition. System speed is 10 microseconds per character recognition.

Research also continues on fuzzy control systems and biomedical fuzzy systems. I expect the importance of biomedical fuzzy systems to grow rapidly in Japan within the next few years, and FLSI will undoubtedly play a major role in this development.

Tokyo Institute of Technology. Michio Sugeno, well known for his pioneering work, heads the Yokohama laboratory at the Institute. Current research involves fuzzy measure theory and fuzzy integrals (t-integral special cases of Choquet, Sugeno, and Weber); fuzzy control; and a fuzzy computer.

Fuzzy control research on one project, a fuzzy controller for an unmanned helicopter, is particularly challenging. The highly unstable helicopter's behavior can be adequately modeled only if 15 input variables and four output variables, which are strongly interrelated, are considered. Attempts to design a classical model-based controller for this purpose have not been successful thus far. Fuzzy control, on the other hand, seems to work quite well, at least on the basis of simulation experiments. I saw results on video of some simulation experiments conducted under various wind conditions and for various remote-control oral instructions (fly straight, turn left, hover, loud). The performance was impressive.

Experiments with a real helicopter are scheduled for March 1992. The experiments will compare fuzzy control and conventional control, the latter being developed independently at another laboratory. This will be an important test, which, I suspect, will demonstrate the superiority of fuzzy control in this and similar applications characterized by high instabilities,

nonlinearities, and time-varying conditions.

The Sugeno laboratory also researches two fuzzy computer projects. One, a linguistic modeling of images using fuzzy case-based reasoning, aims to develop fuzzy logic technology for high-level image understanding similar to that of humans. The second analyzes natural language in the context of the prospective fuzzy computer to develop methods for linguistic modeling and simulation based upon both numerical and linguistic data.

Hosei University. One of the most active academic groups in the area of fuzzy theory and applications in Japan is housed in the Department of Measurement and Control at the Hosei University in Tokyo. This activity is a result of the long-term leadership of Toshiro Terano (currently the director of LIFE) and the more recent leadership of Kaoro Hirota. The group researches a great variety of applications of fuzzy control. Some examples include the tracing of a randomly moving object, control of a yo-yo, and control of a triple inverted pendulum. Experts generally recognize that appropriate stability analysis for fuzzy control systems is currently the most needed component of the emerging fuzzy theory. Other projects include 3D image reconstruction and pattern recognition.

The group is particularly strong in applications of fuzzy logic to robotics. Its goal is not to develop a universal, easy-to-use robot but rather to implement special, highly complicated (and, in some instances, unusual) skills of humans by robots. Completed implementations include a robot playing 2D ping-pong, Japanese flower arrangement by a robot equipped with the knowledge of a human expert, and Chinese calligraphy by a robot. The group also investigated the problem of learning by robots based on fuzzy models.

This group developed a computer-aided instruction system to teach engineers in various industries fundamentals of fuzzy theory and its existing and potential applications. In one year, various companies purchased almost a thousand copies of the system. This fact indicates, again, how strong the current interest in fuzzy theory is in Japan.

A new video camera with image stabilizer impresses all who have seen it.

Recent products

I attended a joint meeting at the Central Research Laboratories of the Matsushita Electric Industrial Company (MEIC) in Osaka. This comprehensive electronics manufacturer researches, particularly in its Intelligence Sciences Laboratory, the development of products that are easy to use. MEIC is one of the founding members of LIFE.

MEIC has already developed and currently manufactures a number of consumer products that use fuzzy control. They include washing machines, refrigerators, air-conditioning systems, vacuum cleaners, microwave ovens, and video cameras. The fuzzy control of the automatic washing machine, for example, senses the quantity of work load, the fabric type, the intensity and type of dirt, and both the room and water temperature. Based on these sensory data, it adjusts the wash, rinse, and spin cycle times on the basis of six fuzzy inference rules that adequately capture knowledge of an experienced operator.

One fascinating new product is a video camera that not only adjusts zoom and flash automatically but also includes an image stabilizer that significantly reduces the movements of the image often caused by a user's shaking hands. The stabilizer compares pictures taken at two sufficiently close time instants and, using appropriate fuzzy inference rules, judges whether any recognized change in the pictures is due to a moving object in the scene or due to camera movement. It then chooses an appropriate corrective action. The performance of the fuzzy stabilizer is outstanding. *[This camera has impressed all those who have seen it.—D.K.K.]*

The lab currently researches the areas of speech synthesis and recognition; text, graphic, and image recognition; 3D image processing, fuzzy data processing; multistage reasoning and judgment; fuzzy data and neural networks; hypermedia; information structuring and classification; and multimedia conversion and integration. Fuzzy theory is involved in virtually all of these areas.

As any other institution I visited in Japan, MEIC also heavily researches the combination of fuzzy theory and neural networks. Its principal interest in fuzzy neural networks is to use them for determining appropriate membership grade functions (by learning from input-output data) for new products.

[David Kahaner is on assignment with the US Office of Naval Research, FarEast. His comments are his own; they do not express any official policy.]

電脳

Guest Editor's Introduction:

Presenting the Far East Special Issue for 1991

Ken Sakamura

University of Tokyo

For starters, let's take a look at some of the differences between the US and Japan in microelectronics research and development. Worldwide, this industry is following the "borderless" trend, making it increasingly difficult to distinguish what is Japanese from what is American. Companies like Motorola and Texas Instruments have plants in Japan, while NEC, Hitachi, and Toshiba manufacture in the US. In this sense, there are many cases in which both countries compete with each other on the same turf.

A clear difference, however, can be seen in the kinds of users in each country. The military and space industries are strong in the US, but almost nonexistent in Japan. On the research level as well, the military sector plays a major role in the US, whereas research at the civilian level is vital in Japan. This factor greatly influenced the style of research and development in each country.

While US companies, of course, consider the needs of the Japanese market, and Japanese companies consider those of the US market, the different needs of their respective domestic markets still clearly influence products. This makes it necessary to be aware of the basic difference in direction of the research and development in each country.

Perhaps due to the relatively large weight of military and space applications, US research tends to reach for extreme limits. Research into real-time processing is a good example. US developers pursue considerable research on "hard real-time" processing, with extremely severe time requirements. The studies are premised on large-scale multiprocessor architectures and large-capacity, ultra high-speed memory. Even if the operating system kernel is compact, the operating system overall is allowed to assume mammoth proportions with scarcely a second thought given to its size.

In Japan, by contrast, where the emphasis is on embedded systems for civilian use, developers typically aim to derive suitable real-time performance from limited resources, or to incorporate memory into the system at minimal cost. Each of these emphases is important in its own right, and so the two countries carry out the different types of research and development. This difference in applications between Japan and the US leads to divergent research aims—and thus approaches—so that the results that emerge also differ.

Now for another update on progress in the TRON Project. The ultimate goal of this project is what I call highly functionally distributed systems. The HFDS concept is premised on a future in which our living environment will be filled with objects embedded with microprocessors, sensors, and actuators. These so-called intelligent objects will have the capability to interact with the environment individually. Moreover, these intelligent

0272-1732/91/0800-0012$01.00 © 1991 IEEE

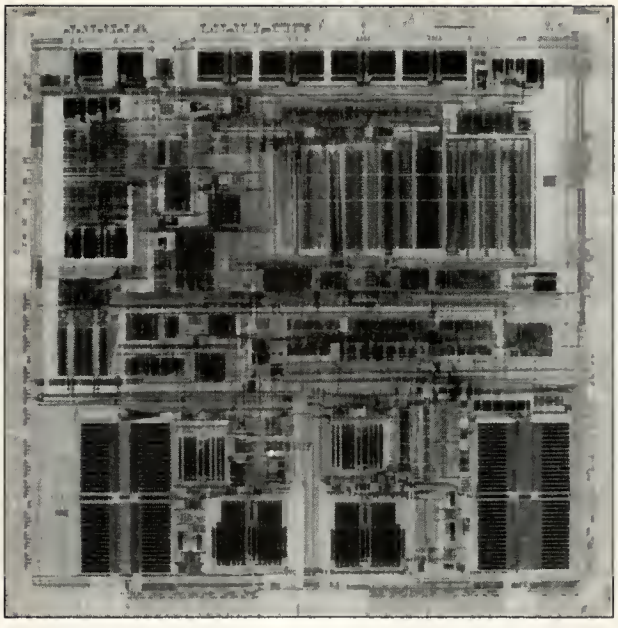


Figure 1. The Gmicro/300 chip.

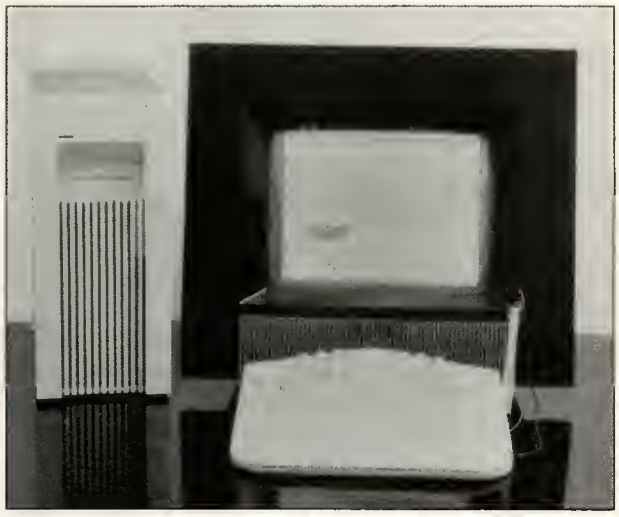


Figure 2. BTRON2-specification computer incorporating the 2B operating system.

objects will be networked together, allowing them to coordinate their actions to some degree. The result will be an extremely large-scale, loosely coupled distributed system, that is, an HFDS.

The TRON Project itself is being carried out on a very large scale. It is divided into two parallel aspects, namely, the foundational projects that define elemental technology from the bottom up, and the application projects that research specific ways of applying the technology from the top down.

As evidence of the progress in the foundational projects, the Gmicro family of TRON-specification 32-bit microprocessors has evolved to the level of the Gmicro/300, which is now being shipped (see Figure 1). This microprocessor achieves 32-MIPS performance when operating at 33 MHz and fully matches RISC (reduced instruction-set computer) performance at the same clock frequency. Development is now continuing on the Gmicro/400 in the 50-MIPS class, the Gmicro/500 in the 100-MIPS class, and beyond that, the Gmicro/600. The TRON-specification microprocessor features a nonproprietary instruction-set specification open to use by all; moreover, it is designed for ready expansion to a full 64-bit architecture.

The ITRON project develops real-time operating system specifications for embedded systems. Here the ITRON2 specification for 32-bit processors and the μITRON specification for 8-bit and 16-bit applications have been followed recently by release of the ITRON-MP specification designed for shared-memory, multiprocessor uses. Operating system development based on this latest specification is already under way. This lineup makes it possible to optimize operating system implementations to the particular architecture, within the same general series.

ITRON specifications are also offered in the public domain for use without licensing requirements, and in Japan operating systems based on ITRON are fast becoming the standard operating system for real-time applications. Meanwhile, several firms are carrying out development based on the CTRON specifications for communication and server operating system interfaces; these are starting to be adopted in PBX products and even in ATM (asynchronous transfer mode) systems for broadband ISDN network switching.

In the BTRON project producing operating system specifications for personal workstations, the latest implementation is 2B, an operating system based on the BTRON2 specification, which was designed especially for TRON-specification microprocessors. (Figure 2 shows a 2B system.) The internal architecture is that of a distributed operating system, with multimedia applications taken into consideration at the operating system design stage. The operating system core has been implemented very compactly based on the ITRON2 specification. (See the BTRON box on the next page.)

TRON-design keyboards (such as the one shown in Figure 3) have started to appear on the market for use with popular personal computer models, and are being acclaimed for a contribution to an improved human interface. Since the public may claim free access to TRON specifications on a nonproprietary basis, six different firms have produced CPU implementations, and many more offer products based on ITRON and CTRON specifications. Validation services

The 2B BTRON-specification operating system

The 2B operating system differs from commonly used text-based operating systems like DOS or Unix in that it assumes multimedia processing from the operating system design stage. It is a multimedia real-time operating system specifically designed to handle graphics, video, and sound at the operating system level. This operating system adopts a standard multimedia-oriented data structure called TAD (for TRON application data bus), which achieves data compatibility even for media other than text.

The human/machine interface and data structure adopted for 2B are inherited from the earlier BTRON1 specification designed to run on the Intel iAPX286 processor. (See "An Overview of the BTRON/286 Specification" in *IEEE Micro*, Vol. 9, No. 3.) The internal structure, however, has been modified for application to larger scale hardware like the TRON-specification microprocessors.

The BTRON2 specification forms the basis for the 2B design, which features advanced modularization of its internal structure and a three-level hierarchy consisting of an inner kernel, outer kernel, and shell. The inner-kernel specification adopts a subset of the ITRON2 specification for embedded-system, real-time operating systems. The outer kernel consists of independent managers for different objects such as files, memory, semaphores, and queues. This flexible design eases maintenance and expansion.

The BTRON2 specification supports applications in distributed environments. The layering and modularization of the internal structure support the distributed operating system functions. Developers use the BTRON1 specification, incidentally, not only in desktop personal computers but also in notebook and other portable computers.

verify conformance to the specifications. Multivendor development is thus taking place on a large scale.

In the application projects, construction will begin soon on a TRON-concept building adopting TRON Project technology throughout, from office automation to building control. We are now evaluating the TRON-concept house, introduced in last year's special issue, by means of actual live-in experiments. Meanwhile, basic planning proceeds on the construction of an entire computer city, which will apply TRON-based technology at all levels.

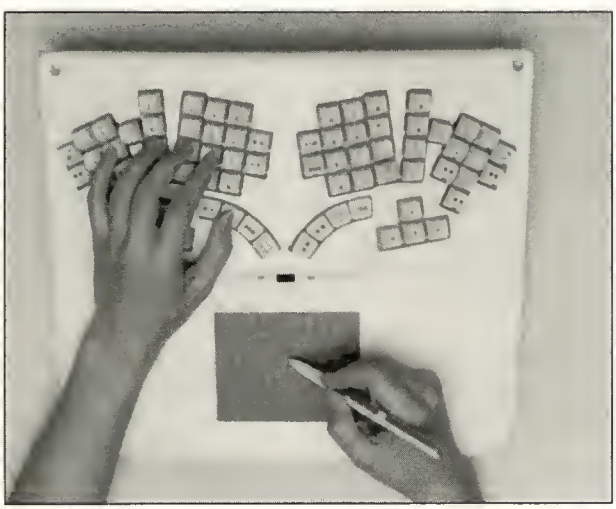


Figure 3. The TRON-design keyboard.

The presentations in this year's special issue are divided in half between academia and industry; those from industry, in particular, can be seen as classic examples of the Japanese style of research and development touched on earlier.

Enhancing microprocessor performance further—assuming general-purpose application is the aim—means either raising clock speed or turning to parallel processing. Since clock speeds are starting to approach their limits, the main approaches today involve either superscalar and superpipeline architectures that increase parallel processing inside the CPU, or else multiprocessor configurations. "Toward Advanced Parallel Processing," contributed by Kyushu University, describes research into raising performance by these approaches.

Another approach raises performance by optimizing for certain types of applications and providing hardware and instructions geared to those applications. This is the approach described in "The Gmicro/100 32-bit Microprocessor" from Mitsubishi Electric Corporation. Designers enhanced performance by applying some ingenious techniques in the basic pipeline design, and also by providing special bitmapping instructions. This approach has for some time been a feature of conventional CPUs; but lately even some RISC processors provide an enhanced instruction set geared to particular applications.

The remaining two contributions describe systems that make use of multiple microprocessors. From the University of Tokyo, "ITRON-MP: An Adaptive Real-Time Kernel Specification for Shared-Memory Multiprocessor Systems" discusses the latest ITRON specification mentioned earlier. This real-time kernel applies to a wide range of multiprocessor systems in shared-memory environments, from hard real-time,

large-scale control systems to large-scale on-line transaction systems.

Finally, "The Architecture of the Sure System 2000 Communications Processor" from Fujitsu Limited describes a fault-tolerant system that uses multiple microprocessors. The architecture employs a number of techniques for improving fault tolerance in both hardware and software. The research examples introduced here may not be as spectacular as what goes on in the US, but I hope they will at least demonstrate the kinds of practical, feet-on-the-ground research and development that is being carried out by large numbers of engineers in Japan today.

Ken Sakamura is an associate professor in the Department of Information Science, Faculty of Science, University of Tokyo. His primary interest lies in computer architecture. He initiated the TRON project in 1984. Under his leadership, over 100 manufacturers now participate in the project to help build computers in the 1990s and beyond. Since he is now also interested in how society will change from the use of computers in the 21st century, his design activities include those for electronic appliances, furniture, houses, buildings, and urban planning.

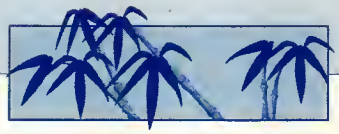
Sakamura received the BS, ME, and PhD degrees in electrical engineering from Keio University in Yokohama. He is a senior member of the IEEE and a member of the ACM. He serves on the editorial board of *IEEE Micro*.

Sakamura may be reached at the Department of Information Science, Faculty of Science, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, Japan; or e-mail at sakamura@tansei.cc.u-tokyo.ac.jp.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 150 Medium 151 High 152



Toward Advanced Parallel Processing:

Exploiting Parallelism at Task and Instruction Levels

Exploiting parallelism at task and instruction levels is the key to advanced general-purpose parallel computing. We are developing a reconfigurable parallel processor system with 128 Sparc microprocessors and a superscalar processor with four operations proceeding in parallel. In the future we plan to replace the Sparcs in the system with the superscalar processors.

Akira Fukuda

Kazuaki Murakami

Kyushu University

Shinji Tomita

Kyoto University

While VLSI (very large scale integration) technology has advanced in recent years, general-purpose computer performance remains static. This fact indicates that designers must exploit other means to achieve future performance gains. We believe the key to future gains in high-performance information processing is parallel processing.

We can look at parallelism from two viewpoints: at the function level or as a computational model. One parallel algorithm implicitly or explicitly comprises the two types of function levels, instruction and task. A computational model includes data parallel processing (SIMD) and control parallel processing (MIMD) models. (See the Glossary for definitions.) In data parallel processing, one operation acts on different data having regular or irregular structures in parallel; a special case is vector processing. In control parallel processing, different MIMD operations proceed in parallel.

To attain high-performance dedicated and general-purpose parallel computers, designers must extract as much parallelism as possible from applications. Many experimental and commercially available dedicated parallel computers have been constructed and work well for particular parallel processing. The parallelism of these computers might be rather easy to exploit. On the other hand, parallelism in general-purpose computers appears much more difficult to exploit, though we feel sure development is necessary to ensure wider use of parallel processing.

Designers can approach general-purpose parallel computers in the following three ways. Let P be the overall system performance level given by $N \times p$, where N is the number of processors, and p is a degree of processing power provided by each element processor.

1) **Today's supercomputer.** This approach corresponds to the case in which N is very small ($1 \sim 4$), and p is very high in performing regular vector data. Performance extends to several Gflops with this approach. However, performance improvement has clearly become saturated, and designers cannot apply the system to a wide variety of applications with irregular data structures.
2) **Massively parallel computer.** In this case, N is extremely large (more than 10,000 processors), and p is very low. The Connection Machine is a typical example of a SIMD system that is very effective for applications with regular data structures. However, MIMD computers of this type have unsolved problems in the algorithmic modeling required to handle massive processors (such as a direct-mapping method), which are qualitatively different from conventional ones. This approach requires a lot of work on this point before we could consider hardware realiza-

0272-1732/91/0800-0016$01.00 © 1991 IEEE

tion. This scheme might become popular in the distant future.

3) **Parallel computers with a medium-to-large number of powerful processors:** Here, N is medium or large (100 ~ 1,000 processors), but p is more powerful than the general-purpose computers (we hope 10 times faster). This approach employs an effective combination of both task and instruction parallelism levels. At the task level, the problems of this approach center on how to divide the original application task into parallel subtasks and how to allocate the subtasks to processors. When the system consists of not too large a number of processors, designers can manage these problems by extending conventional approaches. Interconnection networks and memory structures play an essential role in exploiting task-level parallelism.

   At the instruction level, superscalar and VLIW (very long instruction word) approaches appear promising. Employing low-cost vector units could enhance performance. However, employing various techniques such as software pipelining in VLIW and superscalar processors could provide the same performance level found in low-cost vector units.

We take the third approach. From the viewpoints of today's hardware, software, and application techniques, we believe this is the most promising approach to use in the near future. Since 1986 we have been separately developing both a parallel processor system[1-4] and a superscalar processor[5] to construct a high-performance parallel computer system. That is, the parallel processor system comprises multiple RISC (reduced instruction-set computing) microprocessors (Sun Microsystems Sparcs) rather than the superscalar processors we are now developing. In the future, we plan to replace the microprocessors in the system with the superscalar processors.

This article describes the current status of both projects.

Reconfigurable parallel processor

In a parallel processor system of the above-mentioned scale, the degree to which the architecture matches the parallel algorithms to the applications of interest mainly determines whether the potential performance of the system can be extracted. That is: How well do communication patterns match the interconnection topology of the system? How well do requirements of parallel algorithms for the software memory paradigm match the system's hardware memory paradigm?

Design principles. The key architectural solution to these problems is the reconfigurability of interconnection network topology and hardware memory paradigm.

Network topology. Parallel processor systems with static, fixed interconnection topologies are generally said to have the following limitations: an intractable mapping problem, poor system adaptability, and low program productivity. To

Glossary

Loosely coupled multiprocessor: System in which interconnected stand-alone computers do not depend on each other for multiprocessing and can exchange data on demand.

Message-passing loosely coupled multiprocessor: System having a memory paradigm in which processors access system memory that is partitioned into a set of memory modules associated with the processors. Only the corresponding processor can access each memory module. Interprocessor communications are performed only with the message-passing scheme.

MIMD: Multiple-instruction stream, multiple-data architecture* in which multiple processors autonomously execute different instructions on different data

MISD: Multiple-instruction stream, single-data architecture* in which multiple processors execute different instructions on a single datum. This is generally deemed impractical.

Nonuniform-memory access: Architecture in which multiprocessors share memory and implement nonuniform-memory-access times that depend on physical addresses

Shared-memory tightly coupled multiprocessor: System having a multiprocessor memory paradigm in which any processor can access system memory.

SIMD: Single-instruction stream, multiple-data architecture* in which multiple processors simultaneously execute the same instruction on different data

SISD: Single-instruction stream, single-data architecture* defining serial computers

Tightly coupled multiprocessor: System in which linked computers depend on each other for multiprocessing

Translation look-aside buffer: A cachelike memory that holds recently used mappings of virtual addresses to real addresses

Uniform-memory access: Architecture in which multiprocessors fully share memory and nearly uniform-memory-access times for all data

VLIW (very long instruction word): A parallel processor architecture that uses multiple, independent functional units and packages multiple operations into one very long instruction. An instruction must have a set of fields for each functional unit.

*These terminologies are based on Flynn's taxonomy of computer architecture. Flynn classified architectures on the presence of single or multiple streams of instructions and data.[6]

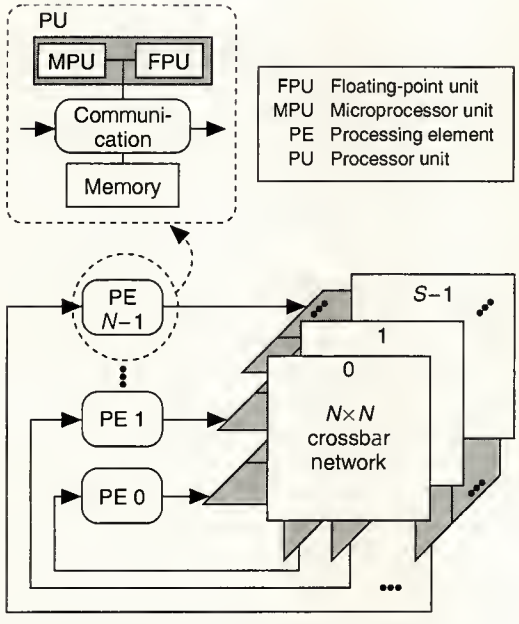


Figure 1. Configuration of the KRPP (Kyushu University Reconfigurable Parallel Processor).

dissolve these limitations, designers can take two approaches. One is to prepare a generic interconnection network within which various communication patterns can be effectively embedded. The hypercube is a typical example; designers can easily embed regular topologies such as mesh, ring, and tree in a hypercube.

The other approach is to tailor the interconnection topology to communication patterns of the application at hand. This approach includes several parallel processor systems with dynamic, flexible interconnection topologies.[7-9] We take this approach because we believe irregular topologies should be effectively realized in increasingly popular applications such as artificial intelligence and neural networks.

Hardware memory paradigm. Applications have different software memory paradigms. Shared memory in a tightly coupled multiprocessor satisfactorily meets the requirements of some applications. Other applications can be partitioned in such a manner that they use only private memory, with minimal message-passing communication among processors; this would result in more efficient execution. A message-passing loosely coupled multiprocessor provides private memory, and an application- and algorithm-dependent approach produces optimal performance. Therefore, we employ a reconfigurable memory architecture. Depending upon application requirements, we can configure the system as a shared-memory tightly coupled multiprocessor, a message-passing loosely coupled multiprocessor, or a hybrid of the two.

Table 1. Specifications of the KRPP.

Item	Specification
Number of PEs (N)	128
Number of crossbar networks (S)	1
Number of switches ($N \times N$)	128 × 128
Inter-PE communication	8-bit-wide circuit switching
Communication bandwidth	Each PE pair: 17 Mbytes/s Total: 2.1 Gbytes/s
Memory organization	Distributed
Memory size	Each PE pair: 8 Mbytes Total: 1 Gbyte
PE configuration	MPU: 16.67-MHz Sparc MB86901 FPU: 16.67-MHz Abacus 3170 Cache: 128-Kbyte SRAM Memory: 8-Mbyte DRAM Message communication unit
Single-PE performance	10 VAX MIPS 1.6 single-precision Linpack Mflops 1.1 double-precision Linpack Mflops
System performance	1,280 VAX MIPS 205 single-precision Linpack Mflops 141 double-precision Linpack Mflops

Our goals for the system are 1) to construct a high-performance, multipurpose multiprocessor system that can be tailored to a broad range of applications, and 2) to offer an experimental environment, or testbed, that encourages many researchers to study highly parallel processing.

System configuration and processing element. As shown in Figure 1, our system, which we call KRPP (Kyuyshu University Reconfigurable Parallel Processor), is a MIMD multimicroprocessor, with N processing elements (PEs) fully connected by S $N \times N$ crossbar networks. The current N and S are 128 and 1 respectively.

The KRPP literally has reconfigurable interconnection net-

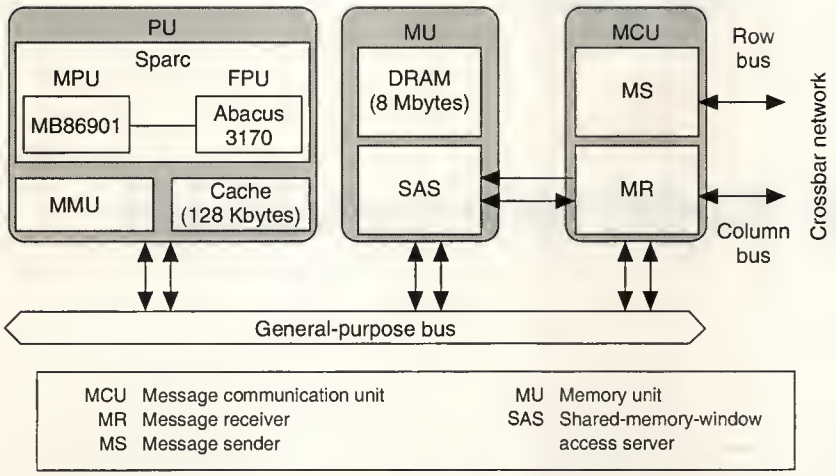


Figure 2. Configuration of a processing element.

works. We define an interconnection network as reconfigurable if its topology can be altered between different algorithm executions or even between different phases of the same algorithm execution. We chose a crossbar network from the viewpoint of performance rather than cost, as described later. The crossbar network allows users to take arbitrary topologies for inter-PE (that is, processor-memory and/or processor-processor) paths under software control of an operating system. Table 1 summarizes the specifications of the KRPP.

Figure 2 illustrates the configuration of one PE. Each PE comprises separate processor, memory, and message communication units, which are interconnected via a single internal bus. The Sparc processor addresses all the units, except for the processor unit and memory, as memory-mapped I/O devices.

The memory management unit in the processor unit contains a translation look-aside buffer and a virtual address cache of 128 Kbytes. The message communication unit provides the network interface and consists of two subunits, the message sender and the message receiver. The memory unit includes a shared-memory-window access server that realizes remote memory references from the other PEs and maintains multicache consistency.

Network architecture. To tailor the interconnection network topology to the communication patterns of applications, we must employ an interconnection network that meets the following requirements:

- It must be feasible to emulate as many topologies as possible. This requirement leads to the following: 1) The network structure should be as flat as possible. If the interconnection network topology reflects a particular topology, the number of emulated topologies would be limited. 2) The routing scheme of the network must be simple. Complicated network routing schemes would need complicated emulation methods.
- It must be feasible to quickly emulate topologies of communication patterns. This requirement leads to the following: 1) The number of network stages should be as small as possible. As their number increases, the access time between processors or processor-memory increases. 2) The degree of network contentions should be as low as possible. A high degree of contentions would increase the transfer time through the network. Nonblocking networks are desirable.

We employed a crossbar network to meet these requirements. It is one of the most powerful interconnection networks in the sense that it is a nonblocking network and provides a path between any PE pair with minimum delay. However, the cost grows to $0(N^2)$ when a crossbar network interconnects N PEs. Therefore, crossbar networks have been considered to be impractical for large systems.

With the technology of large-scale integration (LSI) modularity and high-density printed wiring board manufacturing however, we can now implement a large-scale modular crossbar network with medium-scale crossbar LSI modules.

Configuration and function. Our modular 128 × 128 crossbar network can be derived from 256 identical 8 × 8 crossbar LSI modules, as shown in Figure 3 on page 50. We designed the crossbar LSI module as a CMOS gate array to provide a full crossbar switch between eight ports called X ports and eight other ports called Y ports.

The crossbar LSI module operates in either of the following two modes:

- **Demand.** Each X port independently accepts a connection request from the corresponding message sender in message form and passes it to a target Y port. Since each Y port arbitrates among requests on demand, the switching patterns for the entire LSI module can be dynamically determined through runtime arbitration.
- **Preset.** The crossbar LSI module is microprogrammable to preset 16 switching patterns for every Y port. Every Y port sets up one of those patterns and alternates them periodically and synchronously. Thus we can statically predefine 16 switching patterns for the entire LSI mod-

continued on p. 50

電脳

The Gmicro/100 32-Bit Microprocessor

The Gmicro/100, a 32-bit VLSI microprocessor based on the TRON specification, features a five-stage pipeline to execute instructions. A prejump mechanism implemented as a hardware solution for the jump problem executes benchmark programs 16.8 percent faster on the average. Optimized microinstructions permit bitmap-manipulation instructions to perform two to five times faster than the software loops.

Toyohiko Yoshida

Toru Shimizu

Shigeo Mizugaki

Junichi Hinata

Mitsubishi Electric Corporation

We designed the general-purpose, very large-scale integration Gmicro/100 microprocessor as a core for high-performance, application-specific chips. The TRON specification[1] — which designates a family of architectures, operating system kernels, and CPU chips—provides the foundation for the 32-bit microprocessor.

To accommodate our performance goals, we equipped the Gmicro/100 with a five-stage pipeline that executes basic instructions at 33-MHz clock speeds. Since memory bus bandwidth limits processor performance, we balanced the performance between the memory bus and the execution unit. If we had included a larger on-chip cache, we could have designed the pipeline stage time to operate at one clock cycle. But the chip size requirement for a core processor limited the cache size to 256 bytes. So the pipeline stage executes in two clock cycles. All register-to-register or memory-to-register simple instructions and register-to-memory move instructions execute in two clock cycles. To reduce further pipeline delays, we implemented the prejump mechanism as a hardware solution of the jump problem.[2] By using the prejump mechanism, the Gmicro/100 executes benchmark programs an average of 16.8 percent faster than when we do not use it.

Our secondary objective involved designing the microprocessor to perform the bit-block transfer function at the maximum memory bus speed. The bit-block transfer function allows bitmap display control of personal workstations and laser-beam printers. The Gmicro/100 includes three bitmap manipulation instructions to support bit-block transfer primitives. We implement bitmap instructions as a result of pipelining microoperations of the nth and $(n+1)$th cycles of the microprogram loop. Each microinstruction executes two or three microoperations in parallel. So, the bitmap instructions are two to five times faster than repetitions of simple macroinstructions by the software.[3] At 33.3 MHz, bitmap copy functions perform at a maximum of 267 Mbits/s; Boolean operations reach a peak of 178 Mbits/s.

Overview

The Gmicro/100 includes a 156-Kbit ROM within its 343,000-transistor integration.[1,4-6] The original 25-MHz version—integrated onto a 11.47 ×8.89-mm^2 chip in a 1.0-micrometer, double-metal CMOS process—operated at a 25-MHz clock rate. A 9.17 × 7.16-mm^2 chip contains the shrink version in a 0.8-micrometer, double-metal CMOS process and runs at a 33.3-MHz clock rate (see Table 1). The chip features a 256-byte, direct-mapped instruction cache; a 32-bit data bus; and

0272-1732/91/0800-0020$01.00 © 1991 IEEE

a 32-bit address bus. The chip includes seven major units:

- The instruction fetch unit generates the address of the instruction to be prefetched, includes a 256-byte instruction cache and a 16-byte, prefetched instruction queue.
- The instruction decode unit decodes instructions outputted from the queue.
- The program counter (PC) calculation unit consists of a 32-bit adder and the prejump mechanism. The prejump mechanism includes the branch prediction table (a 256-bit table of the branch histories of conditional branch instructions) and the PC stack (an eight-word stack memory that holds the copy of return target addresses from subroutines). The 32-bit adder calculates the branch target instruction address when the prebranch is taken.
- The micro-ROM unit controls the execution unit according to the microprogram. The ROM chip consists of 111 bits × 1,408 words. Bitmap manipulated instructions use one fifth of the microwords.
- The execution unit consists of execution circuits, a register file, and the communication paths between them. The execution circuits include a 32-bit ALU, a barrel shifter, a priority encoder, and counters. The register file includes sixteen 32-bit, four-port, general-purpose registers (one port connects to the address generation unit, the execution unit uses the other ports) and 10 three-port working registers. Each microinstruction performs one register-to-register operation in two clock cycles.
- The operand address generation unit generates the memory operand address by using a three-input, 32-bit adder. The address generation including memory indirect addressing takes place in this unit independently of the execution unit.
- The bus interface unit controls the address bus and the data bus. Its two-entry operand queue prefetches memory operands before the processing of the execution unit. The bus interface unit also has a one-entry store buffer to concurrently execute an instruction and memory store operation of the prior instruction.

Table 1. Gmicro/100 features.

Features	25 MHz	33 MHz
Pipeline	5 stages	5 stages
Peak MIPS	12.5	16.7
Bitmap manipulation (peak performance)		
Move (Copy)	200 Mbits/s	267 Mbits/s
Boolean function	133 Mbits/s	178 Mbits/s
Instructions	92	92
Instruction cache	256 bytes	256 bytes
Transistors	343,000	343,000
Technology	1.0-µm CMOS	0.8-µm CMOS
Die size	11.47 × 8.89 mm²	9.17 × 7.16 mm²
Package	135-pin PGA 152-pin QFP	135-pin PGA 152-pin QFP

Pipeline scheme

The instruction execution pipeline of the Gmicro/100 consists of five stages (see Figure 1).

Instruction fetch. The IF stage prefetches instructions from the instruction cache or the off-chip memory. Fetching a 4-byte instruction code from the instruction cache requires one clock cycle. In the case of a cache miss, an additional clock cycle is needed. The instruction queue holds prefetched instructions.

Instruction decode. The D stage in the pipeline decodes instructions. Short-format instructions (such as instructions with a one- or zero-memory operand) take two clock cycles to decode; long-format instructions (such as instructions with two memory operands) need twice as long to decode. Two internal codes divide long-format instructions. The following pipeline stages handle the internal codes. The internal codes processed in the pipeline are simple and uniform regardless of the instruction format. So, the hardware of each pipeline stage is simple and the pipeline stage time is short.

Operand address generation. This stage, called A stage, involves two kinds of processing: address generation of a memory operand and secondary decoding of the operation code. The A stage, under hardwired control, handles the address generation, including memory-indirect addressing. Address generation control occurs independently of the microprogram. The hardware interlock mechanism with

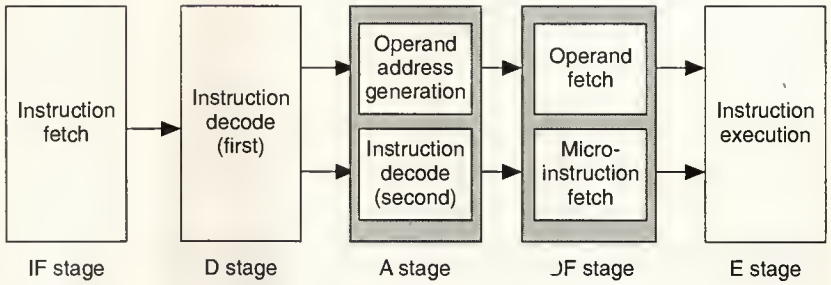


Figure 1. Gmicro/100 pipeline scheme.

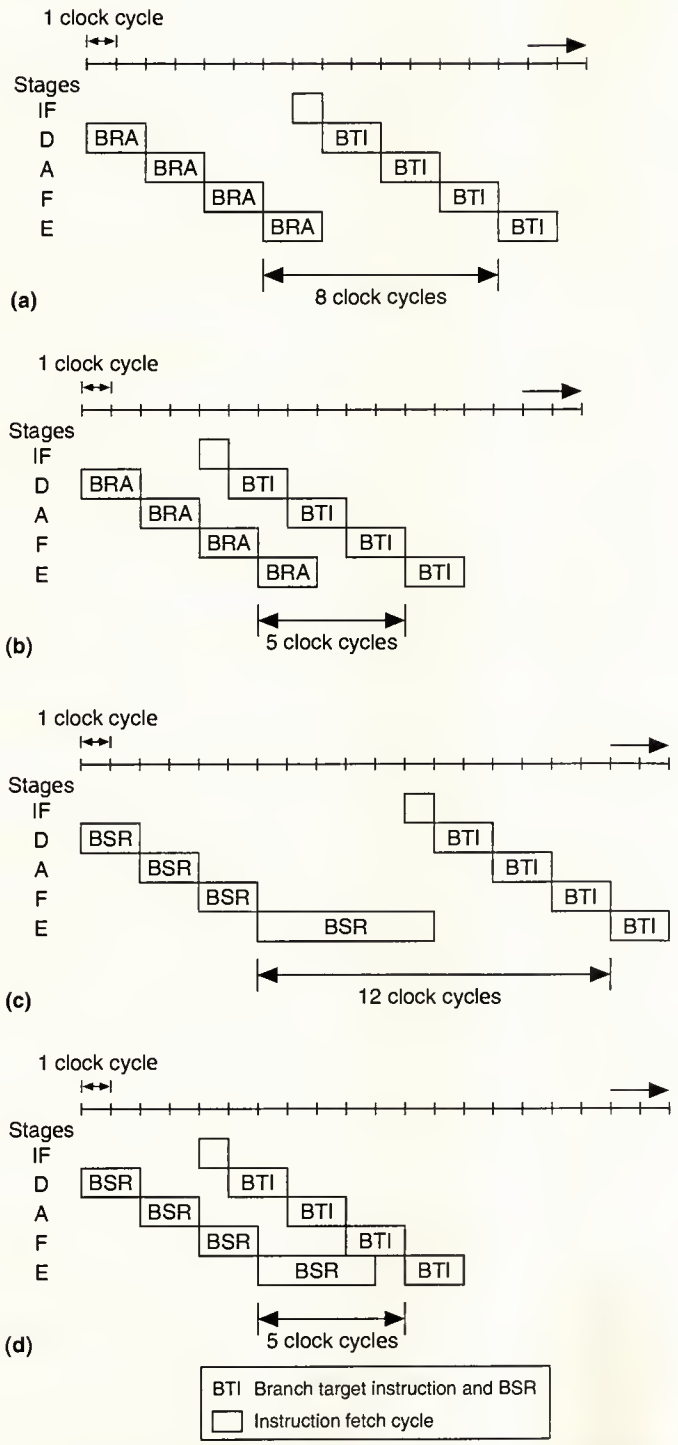


Figure 2. Pipeline time sequences: BRA without the prebranch (a) and with the prebranch (b); BSR without the prebranch (c); and with the prebranch (d).

scoreboard registers checks the data hazard and controls the pipeline flow in this stage.[7]

Operand fetch stage. Also called F stage, this stage performs two kinds of processing: prefetching of a memory operand and a microinstruction fetch from the micro-ROM. F stage is also processed under hardwired control.

Execution stage. E stage executes the instruction based on the microprogram. One microinstruction performs simple register-to-register operations such as move, compare, add, and shift in two clock cycles. Fetching of the memory operand used for the operation has already occurred, and the store-to-memory operation uses a store buffer.

A memory read/write operation performed by the 32-bit address bus and the 32-bit data bus requires two clock cycles. So, we designed the basic pipeline stage time of the D, A, F, and E stages to execute in two clock cycles. When the pipeline is fully operational, all register-to-register and memory-to-register simple instructions and register-to-memory move instructions execute in two clock cycles.

Prejump mechanism

If a jump instruction takes a jump in the E stage, the Gmicro/100 must flush all the stages and fetch instructions from the new path. This flushing and refilling reduces the performance of the pipelined processor. We call it the branch problem. Numerous approaches have been proposed to overcome the branch problem,[2,8] but difficulties involved in the return-from-subroutine instructions are not easy to solve.

The prejump mechanism reduces the penalty not only of branch instructions but also of return-from-subroutine instructions. The prejump occurs in the D stage (the second pipeline stage). This mechanism reduces the penalty for unconditional branches (including the branch to a subroutine), conditional branches, and return-from-subroutine jumps. The prejump taken by unconditional branch instructions and conditional branch instructions is the program counter relative jump (the prebranch). The prejump taken by return-from-subroutine instructions is the memory indirect jump with a stack-pop addressing (the prereturn).

Unconditional branch instruction. The prebranch taken by an unconditional branch instruction is always correct. The prebranch mechanism for an unconditional branch instruction is well-known[9,10] and costs an additional 32-bit adder. When the D stage decodes the unconditional branch instructions, such as the branch always instruction (BRA) and the branch-to-subroutine instruction (BSR), the PC calculation unit calculates the branch target address instead of the next instruction

address. Also, the sequence of the program flow is changed to the branch target instructions. Figure 2 shows the pipeline time sequences of the BRA and BSR instructions and their branch target instructions.

The BRA instruction executes in five clock cycles when it takes a prebranch in the D stage, eight clock cycles if it takes a branch in the E stage. The prejump mechanism restores three clock cycles for the execution of the BRA instruction. The BSR instruction executes in five clock cycles when it takes a prebranch in the D stage, 12 clock cycles when it takes a branch in the E stage. The prejump mechanism restores seven clock cycles for the BSR instruction.

Conditional branch instruction. Taking the prebranch for a unconditional branch instruction is easier than for an conditional branch instruction. The reason is due to the conditional branch instruction in the D stage, which is not capable of determining whether it will take the branch before the completion of all the previous instructions that change the flags in the processor status word.

The microprocessor contains a 256-bit table (the branch prediction table) of the branch histories of conditional branch instructions.[2] The table holds 256 one-bit branch histories, and the hardware modifies them automatically depending on the runtime executions of conditional branch instructions. When the D-stage receives a conditional branch instruction as an input, the stage also gets the branch prediction table content (or the prediction bit). We decode the instruction as a prebranching instruction or a nonprebranching instruction, depending on the prediction bit. The lower eight bits of an instruction address is used to access the table. The prediction table is accessed by using the address of the instruction that is decoded immediately prior to the conditional branch instruction.

Gmicro/100 ASSP terms

Bus arbiter: Arbitrates bus accesses between the Gmicro/100, a direct memory access controller, a refresh controller, and an external bus master.

Direct memory access controller (DMAC): Provides four independent DMA channels. The maximum rate of memory-to-memory transfer is 20 Mybtes/s at 20 MHz.

DRAM refresh controller: Generates refresh addresses and refresh interval pulses for the DRAM.

Interrupt controller: Controls 14 external interrupts and generates an EIT (exceptions, interrupts, and traps) vector number.

Microprocessor wait-state controller: Controls the number of wait cycles according to the monitored address generated by the Gmicro/100 or on-chip DMAC.

Timer: Composed of one 8-bit prescaler and three 16-bit counters. Each counter has a dedicated terminal for output.

Universal asynchronous receiver and transmitter: A fully duplexed and double-buffered serial communication channel.

We use this new method to fetch the prediction bit before the decoding of the conditional branch instruction. The lower eight bits of the instruction address that were executed in the E stage are used to access the table when the following conditional branch instruction takes a branch (it means the prediction is not correct). When a conditional branch instruction is decoded as a prebranching instruction, the PC adder calculates the branch target address. In addition, the operand address calculation adder calculates the address of the next instruction of the conditional branch instruction preparing for the incorrect prediction. Figure 3 depicts the hardware of this mechanism. The hardware exchanges the code length of the conditional branch instruction (A) and its branch displacement (B) when the prediction bit means that the prebranch should be taken. The hardware

continued on p. 62

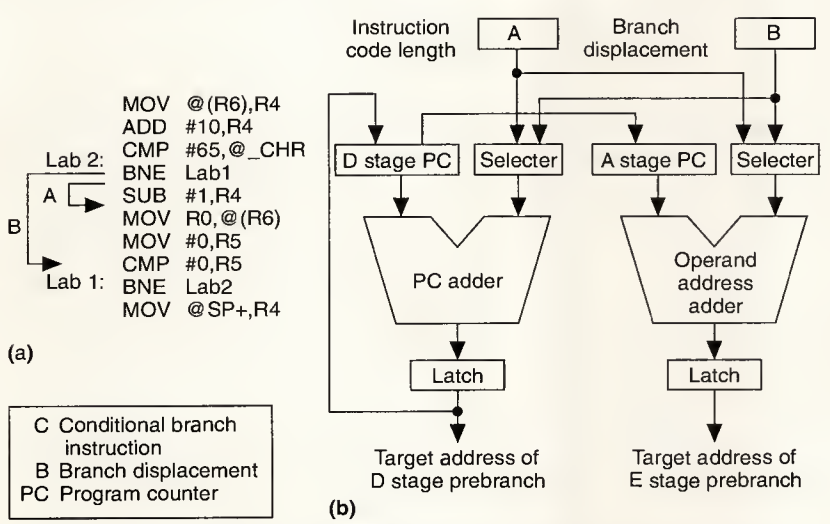


Figure 3. Prebranch mechanism: a portion of compiler output code for the Dhrystone benchmark (a) and a diagram of the instruction code length and branch displacement (b).

電脳

ITRON-MP:

An Adaptive Real-Time Kernel Specification for Shared-Memory Multiprocessor Systems

Designers can tune the ITRON-MP multiprocessor architecture, an adaptive real-time kernel specification, to various shared-memory multiprocessor architectures. ITRON-MP serves as a standard real-time kernel interface that reduces system development.

Hiroaki Takada

Ken Sakamura

University of Tokyo

As the application areas of embedded real-time systems expand, requirements for large-scale and high-performance real-time systems increase. Areas of rapid growth include large-scale control systems (plant- and aircraft-control systems), transaction processing (on-line banking and seat reservation systems), and communication processing (packet switchers and network routers).

In these application areas, a system requires not only large amounts of computational power but also responsive interrupt service and fast task switching in microseconds to handle a large number of external devices. Because one processor cannot satisfy these requirements, multiprocessor systems are emerging as a solution. A multiprocessor system is a particularly effective approach in making a system responsive to many external events.

Though the design and implementation of several real-time operating systems for multiprocessors meet these requirements,[1,2] designers choose to develop most systems for a specific machine or an architecture model. The specification of such an operating system usually depends on the architecture of the machine. However, in a multiprocessor architecture, processor connection topology varies widely with the property of its application. Thus, an operating system designed for a specific architecture cannot be used as a general-purpose one. It is currently necessary to implement an operating system for each machine or develop an application system without any operating system. In such a situation, the lack of a standard development process of application programs brings down the efficiency of software development. This problem is more serious with an embedded real-time system, because new hardware is usually designed for such a system.

We present a real-time kernel specification named ITRON-MP (Industrial-oriented TRON for Multiprocessor Systems), which defines a standard kernel interface adaptable to various shared-memory multiprocessor architectures. ITRON-MP is an extension of ITRON, which is a specification of real-time kernels for embedded systems and one of the subprojects of the TRON project.[3,4] (The TRON project aims to establish a computer system architecture; the project receives support from more than 100 European, American, and Japanese companies). The ITRON-MP specification is an open architecture, in that anyone can freely implement a kernel based on the specification.

A universal implementation of ITRON-MP generates an adapted kernel code for each architecture. This approach presents a standard operating system interface for the development of various real-time systems and improves the efficiency of system development. Moreover, because we tune the executed kernel code to each architecture, we do not degrade the runtime performance by the standardization.

0272-1732/91/0800-0024$01.00

Design concept

We first introduce the concept of the adaptation mechanism of ITRON-MP and then describe the design goals and approach.

Adaptation mechanism. As the most important feature of ITRON-MP, the adaptation mechanism realizes the adaptability to both an architecture and an application.

First, the adaptability to an architecture allows the use of a kernel based on the ITRON-MP specification with various multiprocessor architectures in spite of their differences. These differences include the kind and the number of processors, processor connections, and the accessibility of hardware resources from each processor. Because the design of the architectures for embedded systems must be optimal for each application, we cannot ignore the differences among them.

Figure 1 illustrates typical examples of multiprocessor architectures. In a symmetric architecture (Figure 1a), every processor has equal access to each hardware resource. Commercial multiprocessors widely adopt this architecture because of its generality. On the other hand, asymmetric architecture (Figure 1b) involves the diversity of hardware resource accessibility from the processors. The advantage is that reductions in data traffic on the shared bus occur when communication between an I/O device and a specific processor is tight. A heterogeneous architecture (Figure 1c) consists of different kinds of processors for different purposes. A massively parallel multiprocessor system often adopts the architecture that takes advantage of the communication pattern of its application. A

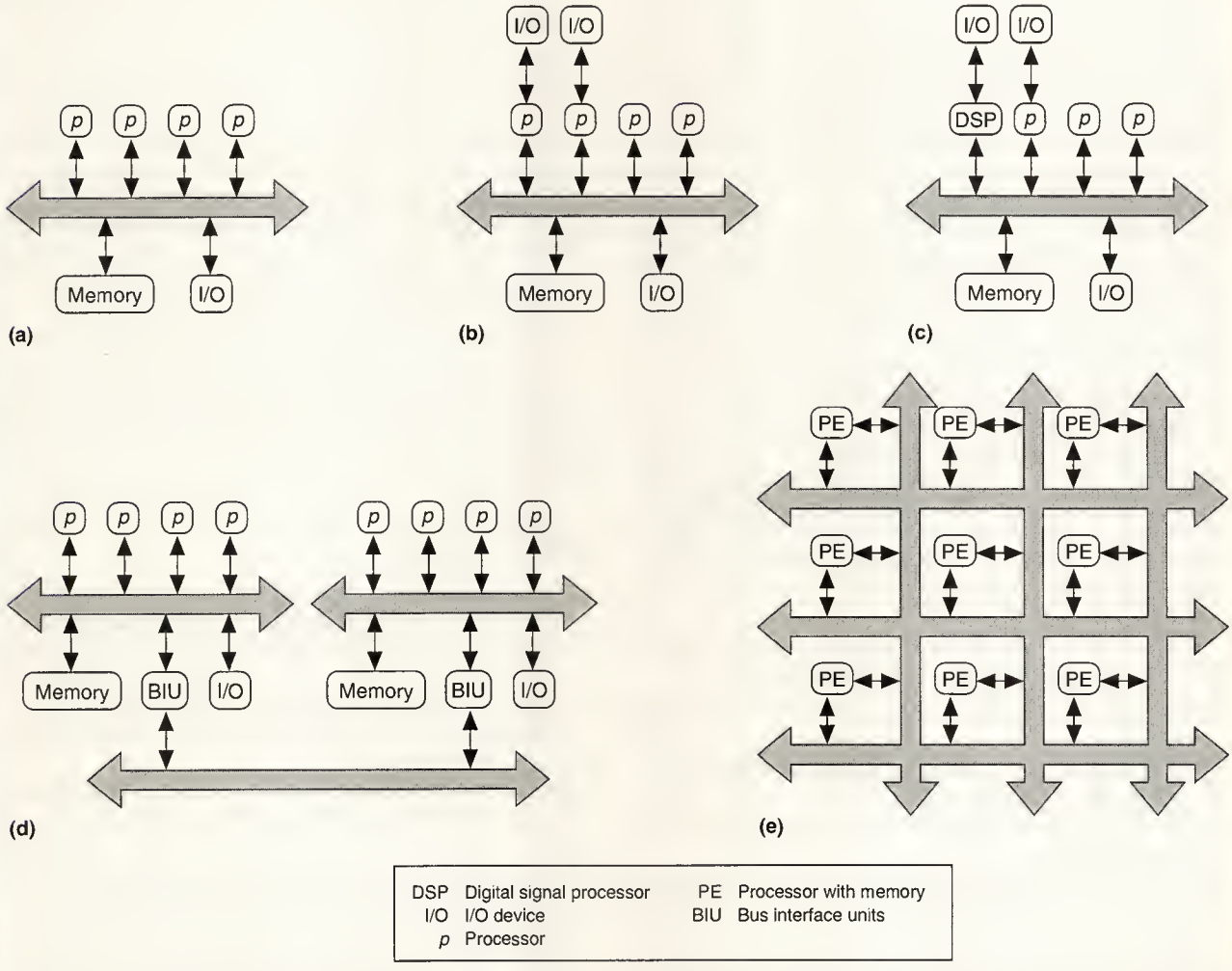


Figure 1. Examples of multiprocessor architectures: symmetric (a), asymmetric (b), heterogeneous (c), hierarchical (d), and array bus (e).

hierarchical bus and an array bus are two typical types of massively parallel architectures (Figure 1d and e). They are other kinds of asymmetric architectures, in that the accessibility of some hardware resources is not uniform.

In the conventional approach, developers design a kernel for each multiprocessor architecture. The difficulty in standardizing the development process of application programs due to the lack of a standard kernel interface and the necessity of implementing an operating system for each machine present serious obstacles for improving the efficiency of software development. To cope with these problems, we adopt the following approach in ITRON-MP. In the specification, we define a standard set of kernel interfaces that can adapt to wide varieties of multiprocessor architectures. We use a tuned kernel code—generated from the architecture's description and the kernel constitution—for the construction of application systems. We call this approach an adaptive operating system.

Next, the adaptability to an application permits the changing of a kernel's implementation according to both the kernel functions and the performance requested by the application to improve the total system performance. This approach is particularly effective for an embedded system, because the kernel code is generated for each application.

For example, when dedicating one of the processors to a heavy load task, we remove the task dispatching mechanism from the kernel for the processor, which improves the runtime performance of the system. The responsiveness of a system is usually not compatible with its total throughput. In this case, it is important to balance the trade-off between the responsiveness and the total throughput for each application.

We avoid cutting the functions of ITRON-MP excessively for simplification, and we offer programmers wide selections of functions. By making the most of its adaptation mechanism to an application, ITRON-MP can make the efficiency of system development compatible with its runtime efficiency.

Goals and approach. The following are the design goals of the ITRON-MP specification:

1) Presents valid approaches for various multiprocessor architectures (adaptability to an architecture);
2) Provides optimal kernel code for the nature of its application (adaptability to an application);
3) Maintains, not degrades, the native performance of a machine or an architecture;
4) Permits easy grasp of the real-time natures of the system developed on it;
5) Applicable to applications requiring fault tolerance; and
6) Allows users to learn it easily.

To achieve these goals, we adopted the following approaches to ITRON-MP. We can meet the first two goals by offering a standard set of kernel functions that can adapt to various architectures and applications. Also, we can offer a mechanism to generate a kernel code containing only suitable functions for each architecture and application.

To achieve the third and fourth goals, we avoid excessive abstraction of kernel resources and, consequently, obtain maximum performance. Programmers can easily write an application with hard timing constraints by being conscious of the underlying execution mechanism of the system. For example, the location transparency of kernel resources is often helpful for the portability of software, but it usually contradicts these goals. The ITRON-MP specification allows programmers to be conscious of the information about the execution mechanism, such as the physical location of kernel resources.

Fault tolerance is another important feature for almost all real-time systems. A leading approach that involves the adoption of multiprocessor architecture for a fault-tolerant system has been researched for a long period.[5] As the actual mechanism to achieve fault tolerance varies for each system, ITRON-MP should serve as a basis for the construction of fault-tolerant systems. Then, the ITRON-MP specification includes some kernel functions necessary for the realization of the fault-tolerant feature. For example, ITRON-MP has a set of system calls that permit user programs to take a snapshot of a task and to resume the task from the snapshot. In other words, ITRON-MP should be adaptable to a fault-tolerant architecture.

We also attach importance to the education of application programmers. Naming of the system calls and the parameters of ITRON-MP must be consistent and understandable. Moreover, because few programmers are familiar with the programming of multiprocessor systems, we plan to prepare a collection of guidelines for multiprocessor system programming that improves the software portability and reduces program errors.

Applications. ITRON-MP can operate in the following application areas:

- Large-scale control systems, such as plant- and aircraft-control systems, require large amounts of computational power. They also require hard real-time property, which means that a missed timing constraint can cause catastrophic results. Some systems that control quickly working objects require interrupt response and task switching time in microseconds.
- Transaction processing systems (for on-line banking and seat reservations) usually have a number of I/O devices, such as disk units and terminals. The systems require soft real-time property and a total response time for communication between the system and the end user of 100 milliseconds or 1 second. Though very fast response time is not necessary, a very large number of transactions and fast task switching is required.
- Communication processing machines, such as packet switchers, network routers, and communication proces-

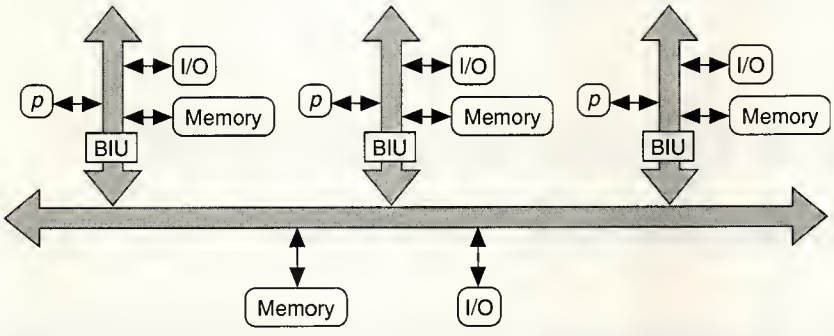


Figure 2. Typical multiprocessor architecture: processor boards with local memory and I/O interface connected to a backplane bus.

sors, serve as the front ends of mainframes. They require fast I/O processing and sometimes need increased computational power for packet routing and so on. Faster processing will be required as multimedia communication becomes popular.

- Special-purpose machines may be dedicated file servers or machines dedicated to a specific numeric calculation or simulation. A file server requires fast I/O processing; a machine for a specific simulation usually requires large amounts of computational power. The massively parallel approach is effective for these kinds of machines. Since connections between processors heavily depend on the application domain, we use the adaptation mechanism of ITRON-MP for this purpose.
- ITRON-MP can also be used as a basis for constructing an operating system with a high-level programming interface.

Main features

There are two design approaches of an operating system for an asymmetric multiprocessor architecture. We can either conceal the asymmetry from the kernel interface or reveal it to the programmers. We take the latter approach so that the programmers can easily grasp the real-time nature of the system. The approach also serves to obtain the maximum performance of the architecture.

Since asymmetric architectures do not permit the accessing of resources, we classify the resources with their characteristics and reflect the classification to the kernel interface. Asymmetric architectures are more general than symmetric ones in this sense, because all resources have the same characteristics in symmetric architectures. We describe the classification of kernel resources in the ITRON-MP specification in detail in this section.

Classification of kernel resources. ITRON-MP makes the accessibility of kernel resources and their access efficiency explicit to programmers according to the principle that the excessive abstraction of kernel resources should be avoided. (Kernel resources include tasks, communication objects such as semaphores and mailboxes, memory pools, and other objects managed by the kernel.) We classify kernel resources by their accessibility, physical location, and other indexes.

In a multiprocessor system, the connection structure among hardware resources (processors, memory units, and I/O devices) determines the runtime efficiency of the system. Therefore, we must classify kernel resources based on the structure of hardware resources used to realize them, so that programmers can grasp the execution time of the system.

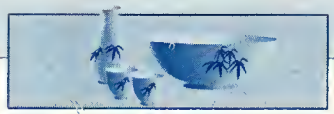
We classify hardware resources according to two viewpoints: physical location and accessibility. For example, suppose a typical multiprocessor architecture in which some processor boards with memory and I/O interfaces on their local bus connect with a backplane bus (Figure 2). From the viewpoint of physical location, we would classify hardware resources as global resources (directly connected to the backplane bus) and local resources of processor p (connected to the local bus of p). From an accessibility viewpoint on the other hand, we would classify hardware resources as *shared* resources (accessible from all processors), and *private* resources of processor p (accessible only from p). As global private resources do not exist, we must classify hardware resources as global shared, local shared, and private local. We simply call these classes global, local, and private, respectively.

An application system that must never stop working uses on-line maintenance technology that allows the insertion and deletion of a board while the system is operational. In this case, we add or remove hardware resources from the system as the unit of a board. Then the physical location of a hardware resource includes two meanings: location from the standpoints of access efficiency and that of replaceable units.

In the previous example, we further divide the global resources, which directly connect to the backplane bus, into the classes corresponding to the boards from the standpoint of replaceable units.

In principle, hardware resources implementing a kernel resource of ITRON-MP should belong to a class of hardware resources. Consequently, we classify kernel resources by their accessibility, location from the standpoint of access efficiency, and location from the standpoint of replaceable units, according to the classification of hardware resources implementing them. The classification of kernel resources represents the kernel model from the programmers' view and does not necessarily exactly agree with the classification of hardware resources.

continued on p. 78



The Architecture of the Sure System 2000 Communications Processor

This processor provides a highly adaptable computer environment and 24-hour-a-day continuous operation. The entire design architecture is based on the local redundancy technique to ensure that no hardware or software fault can cause a system crash. Software errors can be fixed, and hardware can be replaced, upgraded, or added dynamically.

Akira Kabemoto

Hiroshi Yoshida

Fujitsu Limited

More and more users are demanding that their computer systems be as reliable as utilities such as electricity, gas, or water. People want their computers to accept time differences between overseas offices so they can participate more readily in the global trading system, to let them "beat the clock" so they can successfully participate in business opportunities in the exchange and securities market, and to provide extensive, all-night banking services. These needs call for 24-hour, 365-day consecutive operation of computer and network systems that successfully adapt to changing computer environments.[1-3]

Despite these diversified demands, computer system failures in both hardware and software have impeded consecutive operation. It goes without saying that designers must provide reliable systems that can properly cope with faults and failures, thus avoiding system failures and implementing consecutive operation.

Generally though, reliable systems appear most frequently in flight control equipment, spacecraft, and similar applications to support inherent safety requirements. In many of these applications designers duplicate the entire system to incorporate majority logics. For example, three systems process the same job, and the job outputs are compared with each other. When two or more systems' outputs are equal, the output can be used successfully. Duplication, however, inevitably drives the system cost up, making it far too expensive for the general user to accept comfortably. Some mainframe computers and minicomputers feature hot/standby systems for users demanding high reliability. However, they still cost double the conventional systems, resulting in a limited range of application.

Late in the 1970s fault-tolerant computers appeared on the market, targeting general users. As their name indicates, these systems tolerate failures. In other words, they are reliable computers that do not allow the system to fail when a fault occurs and yet carry a relatively low price when compared with conventional hot/standby systems. Some of these computers can remedy both hardware faults and software failures or implement momentary switching control for hardware.

Sure System 2000, a new type of fault-tolerant computer, couples multiprocessors to offer low-priced, high-performance systems that deal effectively with faults and failures.

Existing fault-tolerant computers

Manufactured fault-tolerant computer systems are divided into the two types described in the following paragraphs. They commonly use multiprocessor systems, provide hardware with a particular redundancy for all separate functions, and, when a fault occurs, switch the faulty part to a normal alternative.

The first type of system contains plural paired, redundant processors that connect through dedicated interprocessor data transfer paths. Each processor contains its own main storage. This interprocessor coupling system, which can be considered a loosely coupled multiprocessor in a wider sense, has a higher degree of coupling than general loosely coupled multiprocessor systems connected with external disks or lines because the former type is provided with dedicated paths. In plural paired, redundant processor systems,

0272-1732/91/0800-0028$01.00 © 1991 IEEE

operating systems support each processor. Each of these operating systems performs different jobs and identifies processor faults including software failures by passing checkpoint messages between the operating systems. When a faulty processor is detected, the operating system switches to a good one.

The advantage of this method is that software failures as well as hardware faults can be detected and remedied. However, a directly accessible I/O device can only work on the particular paired processors to which it is connected. Therefore, when users must access a different I/O device connected to another paired processor, the I/O data must be transferred via dedicated interprocessor data transfer paths, resulting in system overhead.

The second type of system contains parallel multiprocessors and shares main storage. Such an interprocessor coupling system, a perfect example of a tightly coupled multiprocessor, inevitably involves a single operating system, which automatically detects a fault in each processor and switches to a spare processor. While a tightly coupled multiprocessor has a high degree of coupling, it can process quickly even with the operating system taken into account, and just as quickly switch processors upon encountering a fault. It cannot deal well with software failures.

As just described, loosely coupled multiprocessor systems can manipulate hardware faults as well as software failures adequately but have a low degree of coupling and other performance problems. Tightly coupled microprocessor systems, on the other hand, perform well but are vulnerable to software failures.

Fujitsu designers aimed to develop a system that satisfies all of the following requirements:

1) resistance to both hardware faults and software failures,
2) high performance, and
3) inexpensive implementation.

This system, called the Sure System 2000, employs multiprocessor systems to extend performance step by step. It features dedicated memory for each processor to troubleshoot the operating system and other software. The operating system works quite independently at each processor.

In addition, Sure System 2000 provides shared memory, equally accessible from each processor, to increase the degree of coupling. The system also duplicates this memory to remedy faults of its own. The shared memory contains data commonly referenced by each processor and recovery data to be used should a fault occur. Already prepared for future address extension, the shared memory contains enough space for a resident database; thus, it can be used as intermediate storage between disks and its own memory.

Processors can transfer data directly to and from each other without using the shared memory. This data transfer scheme allows required local information to be passed between processors.

Very fast common buses interconnect each processor, the shared memory, and each I/O device. Each processor can directly access not only the shared memory but also each I/O device on an equal footing. The common buses allowed a low-priced system to be implemented.

System configuration

Figure 1 shows the logic hardware system configuration of the Sure System 2000. SS2000 modules are generally configured to make the system redundant enough to enable dynamic maintenance. In this way, designers ensured that the hardware configuration can be changed, if necessary, while the system is operating.

The processor module includes a CPU, which contains a Gmicro 300 microprocessor,[4] a TRON-compatible micropro-

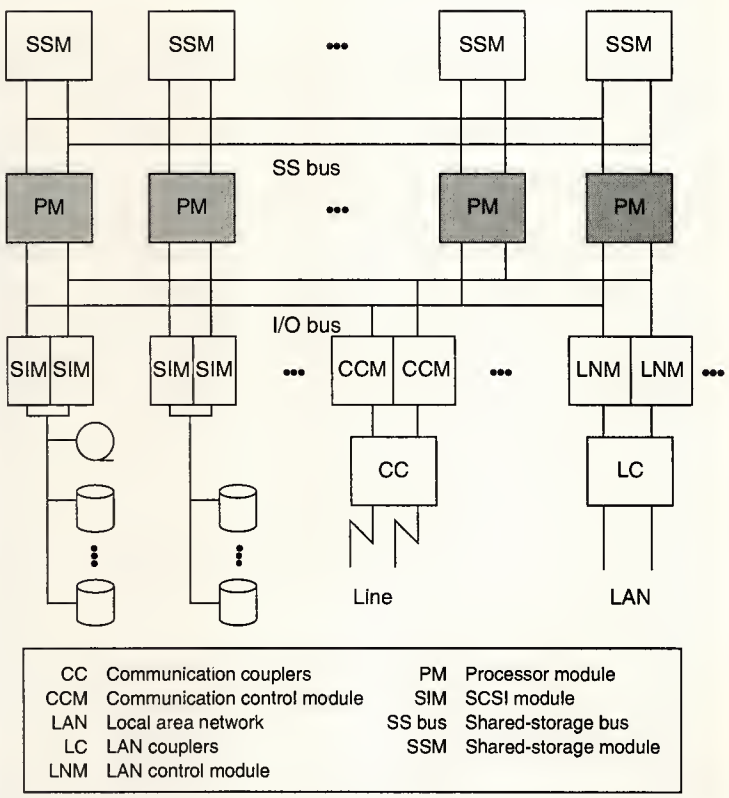


Figure 1. Sure System 2000 configuration.

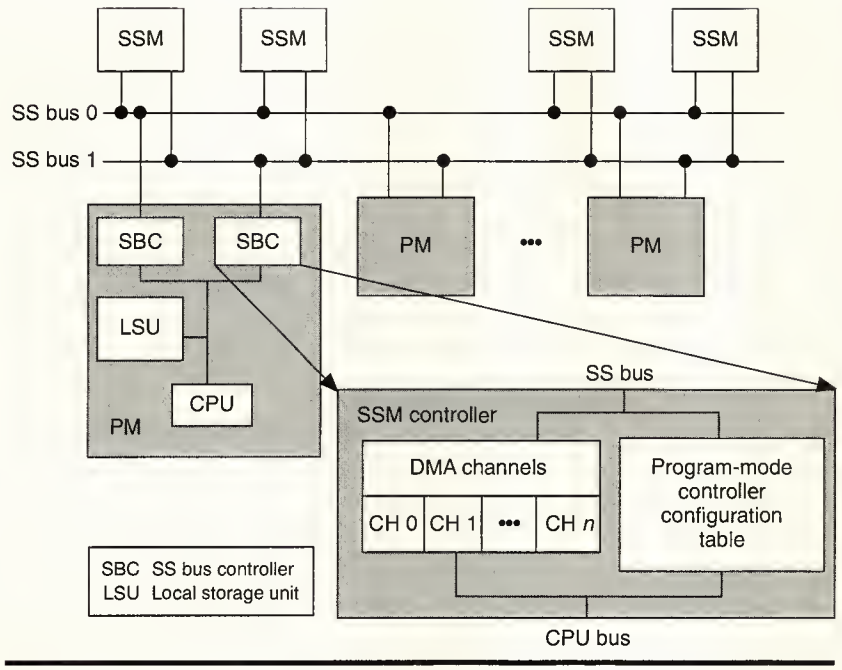


Figure 2. Shared-storage module access.

cessor,[5,6] a local storage unit, and an independent operating system. Therefore, the CPU of one processor module can freely access the local storage unit in the same processor module, but not directly access local storage units in other processor modules.

Each processor module connects to four separate buses. Two buses connect and access shared storage. Two other buses connect the I/O control module group, which drives I/O devices, transfers data among I/O devices and processor modules and among processor modules, and controls the configuration. Duplicating both types of buses makes them fault-resistant.

Processor modules access the shared-storage module on an equal footing. A shared-storage module retains information to be transferred whenever a processor module fault occurs or a checkpoint message is issued.[7]

Modules connected to the I/O bus for I/O control include SCSI (Small Computer Systems Interface), communication control, and local area network control modules. All I/O control modules can be accessed equally from any processor module via the I/O bus.

The SCSI module enables connection of SCSI-compatible magnetic disks and magnetic tapes. The SS2000 incorporates a disk drive and cartridge magnetic tape unit. The module controlling communication lines and the module controlling LANs connect to a variety of communication lines and LANs via different communication and LAN couplers conforming to electrical interface specifications.

Redundant configuration

Each functional unit of the logic hardware of the SS2000 contains individual modules, each of which has a redundant copy. In contrast, in a hot/standby configuration the entire system is duplicated so that, upon failure detection in the hot system, operation switches to the standby system. The system may fail if part of the hot system becomes faulty, thus causing a system switch when a fault is detected in another part of the standby system. However, since each functional unit of the SS2000 contains modules with local redundancy, it can switch a module very quickly without stopping operations, unless multiple failures occur in the same module.

Though all the modules are redundant, they can be used independently as long as they are free of faults. For example, selecting an I/O bus determines the access paths from processor modules to any I/O or line, when different I/O control modules are used. But if no I/O control module is faulty, each I/O control module pursues independent operation and load distribution.

The operating system selects access paths and performs any required switching. Each module retains module states that indicate whether it is normal or faulty. Whenever a module detects its own fault and stops operation, the operating system detects the fault and switches to the standby system.

Processor modules. Processor modules relate on an equal basis and are configured with an N:1 redundancy. Because of local operating system control, the SS2000 continues operating unless all processor modules in the SS2000 become faulty.

Buses and modules. Processor modules connect to two buses. Even if a serious fault occurs on one bus, the other bus can be used to continue processing. The shared-storage module also connects to the duplicate shared-storage buses, so processor modules are immune to faults involving only one shared-storage bus.

I/O control modules operate under processor module control (operating system control). One duplicate I/O control module connects to either I/O bus. Each duplicate I/O module has an interface for accessing the same I/O (disk or line).

If a fault occurs in one of the duplicate I/O buses, all modules connected to the faulty I/O bus become unusable. However, the same I/O is still accessible from the same type of I/O control module connected to the other normal I/O bus.

SCSI modules and disks. The disk is important nonvolatile storage containing frequently used user data. The redun-

dancy designed into the Small Computer System Interface module itself is the same as for other I/O control modules regarding connection to the I/O bus. The paired, redundant modules have a SCSI under their control, thus forming a multiple-initiator configuration. These modules control multiple-disk units, which do not have duplicates. Two sets of paired SCSI modules let disk units connected to physically different SCSIs be duplicated.

This scheme helps guarantee access paths if a fault occurs by implementing multiple redundancy plans. For instance, all disks are still accessible if one I/O bus or one SCSI module becomes faulty.

I/O control modules accept and queue I/O processor module requests. The SCSI multiple-initiator configuration aids operation in two ways. It guarantees access paths if an SCSI module fault occurs, as described. It also improves disk-access performance, distributing the load on the SCSI module and I/O bus by requesting paired SCSI modules to accept I/O processing requests issued at the same time from different processor modules and queuing them through parallel distributed processing.

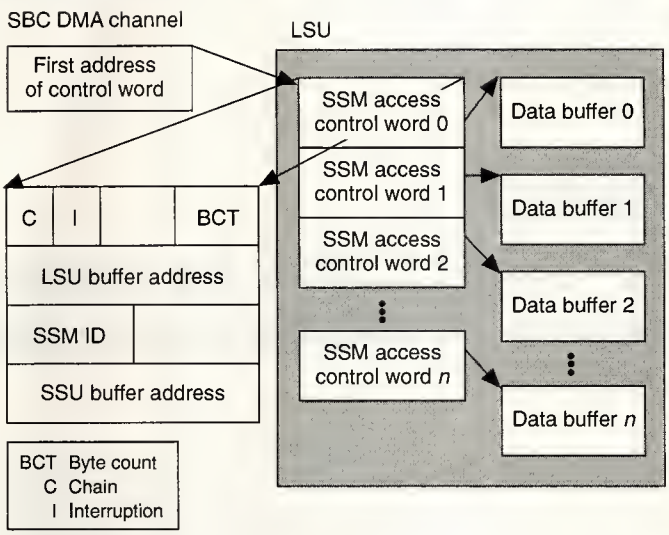


Figure 3. Shared-storage module access in DMA mode.

Multiprocessor architecture

As already stated, the SS2000 can include several processor modules, which differ from conventional tightly coupled or loosely coupled multiprocessors. In tightly coupled multiprocessor systems, the processors share main storage, and one operating system controls all processing. In loosely coupled multiprocessor systems, external disks or lines couple the processors, and each processor contains an independent operating system. Tightly coupled multiprocessors at the kernel level of the operating system differ from the loosely coupled multiprocessors, which are linked at higher levels in the operating system or through applications.

SS2000, together with its operating system, primarily aims to achieve high reliability. Like loosely coupled multiprocessors, each processor module contains its own operating system. Loosely coupled multiprocessors, however, have a lower degree of coupling, so SS2000 designers adopted a shared-memory system, which can commonly be referenced by processor modules. Shared memory cannot contain programs, just data to be inherited or shared.

SS2000 also acquired a higher degree of coupling than general loosely coupled multiprocessor systems. In addition, for local information transferred between particular processor modules, it supports direct data transfer (inter-local storage transfer). These techniques and the supported uniform access to I/O, which will be described later, helped designers to implement an architecture in which loosely coupled multiprocessors can be tightly coupled at the kernel level of the operating system.[8]

Shared-storage module. This module provides two access modes: program and DMA. Program mode enables access to space in the module that is defined in CPU physical address space connected to the CPU. Once the CPU addresses space in the shared-storage module as a source or destination address, general instructions can directly access this space.

In DMA mode, data transfers directly between the local storage unit and the shared-storage module through DMA channels connected to the processor modules without CPU intervention. Each processor module has multiple DMA channels dedicated to access shared-storage modules. Parallel with instructions being executed on the CPU, the shared-storage module is accessed by driving DMA channels under operating system control (Figure 2). This operating system places the local storage unit address, shared-storage module ID, and memory address in the designated module. It also places the length of the data to be transferred in the module access control word, specifying how module access should proceed. It then drives the DMA channel to start the transfer of data between the local storage unit and shared-storage module. Each control word uses chaining and intermediate interrupt reporting to enable data to be transferred without undue restriction (see Figure 3). When access ends, an external interrupt to the CPU notifies the operating system.

An advantage of program mode is that the shared-storage module can be accessed from user space without operating system intervention because of the CPU's dynamic address translation. DAT helps reduce operating system overhead and inhibits incorrect user space access for memory access protection in the DAT table. An advantage of DMA mode is that, when a hardware batch transfer is available, data trans

continued on p. 73

KMDS:

An Expert System for Integrated Hardware/Software Design of Microprocessor-Based Digital Systems

KMDS makes use of a knowledge-based expert system and external algorithmic procedures to realize a modularized and flexible design automation tool. Besides aiding the design of single-board microcomputers, KMDS also helps designers of intelligent interface cards such as color display adapters and printer server cards.

Yau-Hwang Kuo

Ling-Yang Kung

Ching-Chung Tzeng

Guang-Huei Jeng

Wei-Kuo Chia

National Cheng Kung University

Two major considerations dictate today's most popular architecture for digital systems and interface adapters. Because of the rapid progress in IC manufacturing technologies, LSI and VLSI (large-scale and very large-scale integration) devices now constitute the major components of digital systems.[1,2] In addition, users require that digital systems be intelligent, flexible, and secure. Thus, current digital systems generally consist of a microprocessor kernel surrounded with memories, I/O controllers, and application-specific IC chips (ASICs). Small-scale and medium-scale integration devices constitute only a small ratio of the chips in a whole circuit and usually act as "glue" chips.

This trend shifts the design of digital systems from complicated gate network synthesis that intensively minimizes the number of gates to a selection of adequate LSI components for each specified function and then a combination of them. Designers must base considerations of circuit cost, performance, and reliability on a model other than those used for gate-based design. Furthermore, they can raise the input description of a design automation tool to the system requirement level. Under these circumstances, a design often may have many candidates for the microprocessor, and designers must frequently update design knowledge and information. A knowledge-base approach to design may better support designer efforts than the traditional algorithmic approach.[3]

In addition, to combine LSI and VLSI chips, designers must satisfy timing constraints, a critical issue in the synthesis of digital systems. Another important consideration is that, although the work of circuit synthesis may be simplified, the programming required to activate those complex components can form another serious design bottleneck. So, designers must determine the best way to automate the design with an integrated hardware/software design scheme.

We present a design automation system called KMDS that satisfies these goals. A knowledge-based expert system supports KMDS to integrate the following functions:

- automatic design and synthesis of a microprocessor-based digital system from a given user specification at the system level,
- automatic generation of a control program to supervise this system,
- automatic design of an ASIC to implement some special-purpose functions that cannot be realized with existing devices,
- automatic PCB layout and simulation, and
- automatic checking of knowledge-base consistency.

Reasoning in the expert system realizes the first function, automated synthesis, which uses a hybrid frame and rule-based structure[4] to implement the knowledge base. For efficiency, KMDS also

0272-1732/91/0800-0032$01.00 © 1991 IEEE

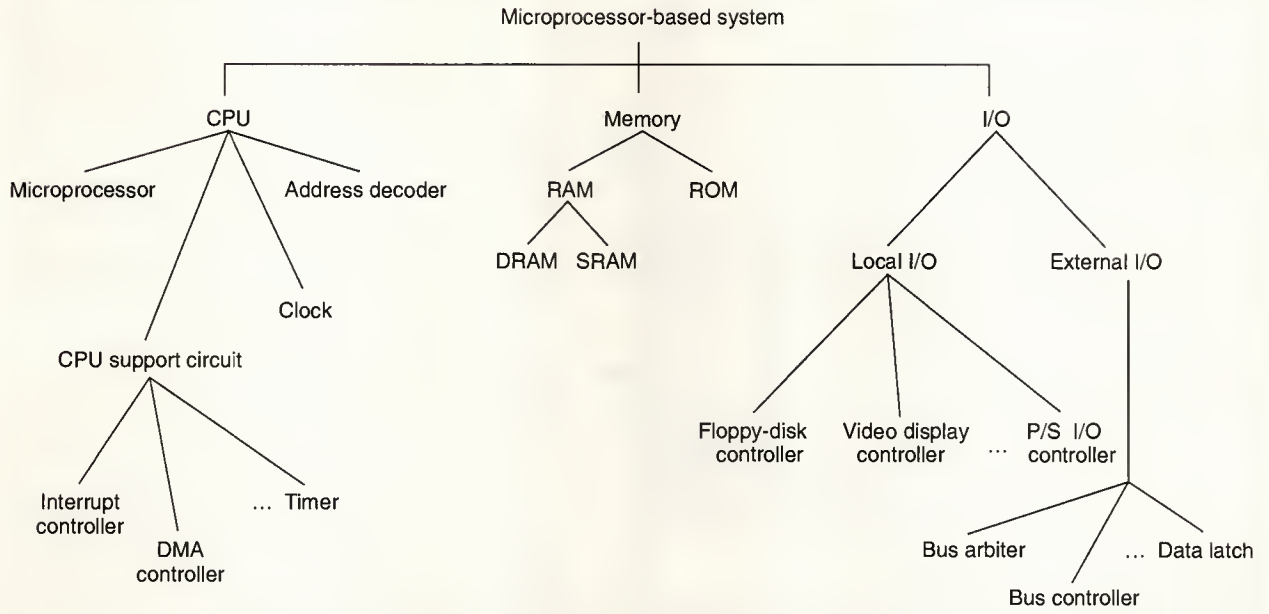


Figure 1. KMDS design scope.

embeds some external algorithmic procedures to be invoked by the expert system.

The second function, automatic control program generation, releases hardware designers from the pain of understanding the detailed programming specification of LSI and VLSI components and from the unnecessary effort of learning every assembly language. In KMDS, the expert system generates the control program of an implemented digital system simultaneously with the hardware into a program written in Emasil.[5] This high-level language describes the behavior and structure of microprocessor-based digital systems. In the transformation process, the control program generator of the expert system gradually introduces machine-dependent parameters into the Emasil program according to the programming specification of selected devices and the user requirements. Finally, a retargetable compiler,[6] combining the advantages of an interpretative method and a table-driven method, translates the Emasil program into the assembly program of the selected microprocessor.

KMDS invokes the third function, automatic ASIC design, when a required circuit function is too specific for the designers to realize it in an existing device. In this case, the user supplies the algorithmic behavior and timing specification of the desired circuit function, then the automatic ASIC design function proposes a petri net-based synthesis technique to implement the circuit function as an ASIC. A high-level tool called the NCKU-DA system[7] realizes this function.

Linking commercial simulation packages (Cadat or System Hilo) and layout programs (HP PCDS) to the expert system implements the fourth function, PCB layout and simulation. As Brown et al.[8] points out, integrating various tools to form a complete environment has become a popular trend. This approach can provide richer functions but need not cause much extra development effort.

Finally, knowledge-base debugging provides expert users with a convenient way to construct, expand, or modify the knowledge base. Three graph-based heuristic algorithms[9] detect circular rule chains, redundant rules, and conflicting rules.

By integrating these functions, KMDS constitutes a complete, flexible, and high-performance environment for the design and implementation of microprocessor-based digital systems.

System configuration and characteristics

Figure 1 illustrates the KMDS design scope as a design automation assistant for intelligent I/O systems and microcomputer systems. Knowledge-based expert system techniques and a set of external algorithmic programs achieve professional quality when synthesizing digital systems.

KMDS realizes the task of digital system synthesis as a problem-solving process; that is, it reaches a synthesis by invoking a sequence of problem solvers. The responsibility of a problem solver is to find a path from some problem state to a solution state or partial solution state. The tools available to the problem solver include a set of operators; the application of an operator transforms the current state to a new state. These operators correspond to the rules in the knowledge base or to an external procedure.

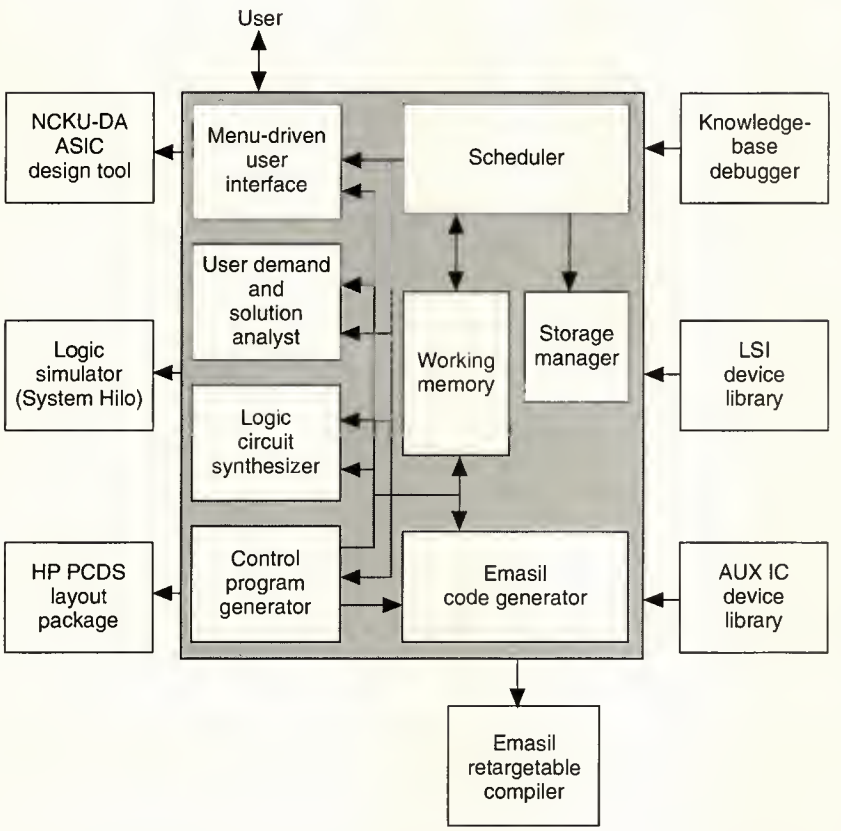


Figure 2. KMDS system configuration.

Figure 2 depicts the KMDS system configuration in which the expert system kernel features:

- a menu-driven user interface,
- a hierarchical and modularized design methodology,
- a modularized knowledge base,
- automatic generation of digital system control programs, and
- a uniform format of hardware and software design knowledge.

The expert system, constructed on a hybrid frame and rule-based structure described later, contains six major modules: the scheduler, menu-driven user interface, user demand and solution analyst, circuit synthesizer, control program generator, and storage manager.

The scheduler controls the design procedure. It switches system control among the other system modules according to the intermediate system working status in working memory. So, the expert system is similar to a blackboard system. The scheduler also controls the invocation of external packages such as an ASIC design tool, retargetable compiler, and so on. Because of this scheduler, the knowledge base can be organized with a highly modularized structure, so that

1) the constraint on the size of working memory is relaxed;
2) the reasoning could be executed on a smaller conflict set, that is, could have a higher performance of reasoning; and
3) the knowledge base is easy to modify, manage, and expand.

During KMDS consultation the scheduler always resides in rule memory. For other system modules, the scheduler will not load the corresponding system module into rule memory unless the status of working memory matches with a specific condition.

The menu-driven user interface provides an easy-to-use scheme for the interaction between users and KMDS. With this interface, a user can describe the circuit specification they want, not the way to design it. Users can select either a mouse or a keyboard as an input device. The interface adopts a form-filling manner to obtain user requirements. For designing a single-board microcomputer, KMDS requires six types of specifications: CPU, memory subsystem, general-purpose I/O, bus interface, DMA, and interrupt control, system enhancement, and design tendency.

On the other hand, KMDS requires four types of specification for designing an intelligent interface card (for example, a color display adapter): interface specification, host connection, peripheral connection, and design tendency. Users can specify their desires concerning four criteria: cost, speed, functionality, and area.

The user interface also supports three services:

- **Checks for requirement reasonableness**. When a user makes an unreasonable demand on the design, KMDS asks the user to modify the requirements. If the user does not, KMDS continues processing. For example, say a user wants two graphic modes to be realized on the designed CRT display adapter: a resolution of 640 dots by 200 dots and 16 colors, and 320 dots by 200 dots and four colors. Obviously, the former mode has higher resolution and more colors than the latter. Although the requirement is not wrong, it is unreasonable.

- **Checks for specification completeness**.
- **On-line consultation**. This function provides information about industrial standards that users may be unfamiliar with and gives some expert suggestions.

The user demand and solution analyst works as a presynthesizer. It transfers a user requirement into a complete design specification and then constructs an architecture frame based on the design specification. When several architectures satisfy the specification, this module chooses the most adequate architecture using an available-factor method, which we describe later. If no architecture satisfies the user's requirements, the user demand and solution analyst will ask for new requirements.

The circuit synthesizer controls the practical hardware synthesis task. It binds major LSI components to the constructed architecture frame and then produces complete and proper interconnections among the components. Finally, the synthesizer generates a netlist file written in EDIF (Electronic Design Interchange Format). This file can be introduced to a commercial logic simulator or PCB layout package capable of accepting an EDIF file.

Besides a hardware circuit, a digital system should have its own control program to direct the activities of the entire machine. Generating the programming needed to activate and control LSI or VLSI devices has become a formidable task as designers integrate more and more functions onto one chip. In addition, because the microprocessor may be chosen from many candidates under the user's specification, the user may not be familiar with the particular assembly language needed to write the control program. Similarly, it is impractical to generate direct assembly programs for all different microprocessors in an expert system. Therefore, the automatic generation of control programs in a unified high-level language is necessary for a fully automated design of microprocessor-based digital systems.

KMDS develops the Emasil language and a retargetable compiler for such a scheme. Emasil allows users to describe a digital system at the processor-memory-switch level. A program written in this language includes not only information concerning the hardware structure but also the interactions between hardware devices (the programming information).

KMDS uses a control program generator to produce the control routines of a synthesized digital system in Emasil. The related hardware information, such as memory space, I/O port address, is also included in the Emasil program. The generator chooses suitable algorithms (represented as frames in a knowledge base) and extracts the component programming factors from hardware frames, according to the user's specification. Then it introduces the programming factors into the chosen algorithms and, finally, packs the algorithms into a complete control program written in Emasil. The retargetable compiler then translates the Emasil program into the machine codes of the selected microprocessor.

Table 1. Slot representation. Frame name: 63484 CRT interface module.

Slot no.	Slot attribute	Value
1	CRTC series no.	(63484PS4/63484PS6/...)
2	CRTC frame	(CRTC63484)
3	Buffer	(FM8/FM16)
4	Data bus width	(8/16)
5	Bits/pixel	(1/2/4)
6	Maximum resolution	(Columns, rows)
⋮		
N	Signal pool	(S1,S2, ...)

The storage manager controls the access of all knowledge modules, the device libraries, and attached external programs.

Knowledge base and device libraries

Because a microprocessor-based digital system itself has the characteristics of hierarchy and modularization, KMDS adopts a frame structure to represent it. Frames in KMDS are organized into trees, with each frame at the lower level inheriting the properties of the frames at higher levels. The two types of KMDS frames, packed and primitive, have a basic format that consists of

1) a name,
2) a parent frame,
3) children frames,
4) slots and their values, and
5) attached predicates.

The main distinction between the two types of frames occurs in the slots. In packed frames a frame includes three kinds of slots: specification, signal pool, and construction. The specification slots contain associated specifications of the hardware module represented by the frame. The construction slots describe the internal structure of the hardware module. The value of a construction slot points to another frame; that is, it specifies a specific activity (submodule) of the frame. The signal pool slot is the collection of all signal information of the frame. Only one signal pool slot exists in a frame. See Table 1 for the representation of some slots of a CRT interface module. In this table slots 1, 2, and 3 are construction slots, slots 4,5, and 6 are specification slots, and slot *N* represents a signal pool slot.

Primitive frames do not contain construction slots. In other words, a primitive frame stands for one primitive functional

continued on p. 86

Europe on the way to the metric system

Hubert Kirrmann

Asea Brown Boveri Research Center

CRBC.1 CH-5405 Baden, Switzerland

It may seem strange, but continental Europeans, who first introduced the metric system (see box), still use inches and feet. This phenomenon finds its tradition in electronics and informatics, where the bulk of publications come from the Anglo-Saxon world. Lately, the workstation-based tools developed for the world market prompted the use of English units and conventions in other fields as well. Designers originally wrote CAD tools, simulators, and drawing programs for the English unit system. As a consequence, many nonmetric habits have spread. Even when the software supports a metric mode, it is often safer to work with inches to avoid rounding errors (see box on standardization of measurement expressions on p. 38).

For computer hardware designers, the Eurocard crate system, known as the 19-inch rack, has become the Tower of Babel of metrology. The mechanical dimensions are a beautiful mix of millimeters and inches. Boards and connectors take steps of 1/10th of an inch (2.54 mm), while their outer dimensions appear in metric measurements. Measurements in inches and millimeters even mix within one dimension: a double-size Euroboard has 160 mm multiplied by (100 millimeters + 5.25 inches), or 160×233.35 mm^2. In spite of this, the 19-inch system is so widespread that the International Electrotechnical Commission (IEC), which is strongly committed to the metric system, standardized it (IEC 97, IEC 326, IEC 297, and IEC 603).[1-7] Even the German standard body, DIN, standardized connectors with 1/10th-of-an-inch steps, such as the popular DIN 41 612 96-pin indirect connector.

In the US and Europe, all recent computer buses adopted the 19-inch racks and connectors: VMEbus, Multibus II, Nubus, and Futurebus. The Institute of Electrical and Electronics Engineers set up its 1101 standard group to specify the application of the 19-inch system to computer buses.

The limitations of the 19-inch system are well-known. Mechanical tolerances are insufficient—especially when using more than one connector—and the unit mix leads to confusion when extending the system. Mechanical designers would really like to let 1 inch equal 2.5 cm, but unfortunately 1 inch equals 2.54 cm. This small difference has far-reaching consequences: Numerically controlled machines, CAD tools, layout programs, automatic inserters, and test nailbeds follow either the inch or metric rastering, but not both at the same time. Although some European and Japanese integrated circuit manufacturers introduced components with a 2.5-mm raster (rather than 2.54 mm), they had little success.

As work on extending the life of the 19-inch system progressed, it became readily apparent to designers as early as 1980 that they could only achieve a technological step by untangling the units. So, the designers developed a whole new electronic packaging system, the Modular Order (MO), on the basis of three-dimensional, purely metric rastering. The German firm, Siemens, was the driving force behind this standard; other firms like Knuerr and Harting also participated.

The basic unit of MO is a unit cube of $0.5 \times 0.5 \times 0.5$ mm. This rastering is used with ICs, boards, connectors, racks, cabinets, and entire electronic rooms. To ease transition from the 19-inch system, MO maps the dimensions using the 1 inch equals 25 mm reference.

MO is much more than a pure metrification of the 19-inch racks. The unification of measurements allows many technological limits to be

0272-1732/91/0800-0036$01.00 © 1991 IEEE

The evolving metric system

It is sometimes hard to believe that the US adopted the metric unit system more than a hundred years ago. At the 1884 Washington Conference, the French agreed to recognize the Greenwich meridian as the longitude reference (in place of their Paris meridian) if the Americans and British would agree to introduce the metric system. After adoption of both agreements, the British representative stood up and declared that "the English goverment does not feel authorized to impose such or such a system for weights and measures to the subjects of the Queen." "Perfid Albion" obviously won the wrong battle.

The defining of the metric system occurred during the wave of the French Revolution, which contributed little to its acceptance. In 1790, the Constitutional Assembly asked the French Science Academy to define a unified system for weights and measures that would take the place of the hundreds of regional units that often shared the same name. Between 1792 and 1799, Mechain and Delambre measured the length of the earth's meridian between Dunkirk, France, and Barcelona, Spain. This measurement served as a basis for measuring the total length of the meridian, which was considered at the time to have the same length as the earth's equator. Mechain and Delambre divided the equator into 40,000,000 parts, and the length of this part received the name of meter.

This choice simplifies navigation. Indeed, the metric system divides the circle in 400 grades of 100 centigrades, one centigrade equaling 1,000 meters. (The nautical mile is based on a similar choice, but the earth's equator is divided into 360 degrees of 60 [sexagesimal] minutes, one minute equaling one mile). Although most professions today use the 360-degree division, the 400-grade division is in wide use in land surveying and artillery coordinates.

Since the earth's equator was not a practical reference, a platinum/iridium ruler, kept at constant temperature, became the standard. At the same time, Lavoisier defined the gram as the mass of a water cube of 1 cubic cm at a temperature of 4 degrees (Celsius).

At the 18th Genminal of the Year Two (April 7, 1795) the French Revolutionary Convention declared these measurement units as mandatory and defined their symbols, multiples, and submultiples as they appear today. Excessive metrifications—like weeks of 10 days, 10 hours, and 100 minutes in length—have since disappeared. It appeared unlikely that the Revolutionary Convention could impose a year of 400 days. Since 1799, a cylinder of platinum/iridium with a mass of 1,000 grams (corresponding to one liter of water or 1,000 cubic cm) serves as the mass reference. Time was defined based on the second—a fraction of the astronomical year.

Not until 1964 did the metric system evolve to become the International System (SI), based on seven basic units: meter, second, kilogram, ampere, mole, kelvin, and candela. The International Standard Organization (ISO) defines the metric system in its ISO 31 standard and offers application guidances as ISO 1000 and IEC 27. Starting in 1960, the meter was based on the wavelength of Krypton 86. Since 1983, the meter has been based on the speed of light in a vacuum, which was defined as 299,792,458 meters per second. The original meter standard ruler is in a museum. A redefinition of the time unit also occurred in 1983; a cesium 133 atomic clock is now the basis of the time unit.

The SI evolution is not finished, however. Indeed, many units are related when they can be derived from another. For instance, the force unit Newton is not a basic unit since it can be reduced to $1\ N = 1\ kgm/s^2$. Now, if the speed of light is a constant, meters and seconds are related; a person needs only to specify one or the other (which is what SI has done recently when redefining the meter.

According to Einstein's relation $E = mc^2$, it would be legal to express energy and mass with the same unit of measurement since the speed of light is a constant (unless somebody proves the contrary). One could write the equation $E = m$, since 1 kg equals $8.9874\ 10^{16}$ joules. In the same line, a whole new unit system has been defined relying only on the time unit and on a set of physical constants, among them the speed of light, the Plank constant, and the Bolzmann constant. However, this system is rather theoretical.

The problem of the unit system is a reliance on a set of measurable references. For instance, although electromagnetism is based on the ampere, the volt proves to be a handier reference. In fact, the new definition of the volt, based on the Josephson effect, could replace the ampere.[8]

A main advantage of a unifying unit system like the metric system is that conversion factors disappear. One can write: Force = Mass × Acceleration, and this expression will be correct if the force is expressed in newtons, the mass in kilograms, and the acceleration in meters per second. Checking the correctness of expressions is very easy. (It also works in the English system if the units are consistent, for instance when inches and feet are not mixed.)

overcome. Designers can modularly extend a whole new system of boards, cartridges, racks, and ventilation slots in all three dimensions. Boards offer an improved form factor: The 265 × 235-mm² board replaces the popular double Euroboard (233.35 × 220 mm²), with a gain of 21 percent of real estate. As a special bonus, both sides of the boards can use surface-mount devices.

The most interesting part of the work involves the connectors. In fact, the major manufacturers developed metric connectors in parallel to the MO effort to overcome the DIN 41 612 limit of 96 pins per connector. This explains why DuPont's Metral uses a 2-mm raster, which is contrary to the 2.5-mm raster used by MO. AMP (Aircraft and Marine Products, The Netherlands) developed a five-row, 2-mm connector based on DIN 43 355, which gives room for 255 pins per 100 mm. On the other side, Siemens, Molex, and Harting developed connectors with 2.5-mm spacings as specified in DIN 41 642. These connectors are very versatile: Power pins, 50-ohm coaxial connectors, and optical connectors can mix with standard and forerunning pins. Connections may be wrapped, soldered, or pressed in. EMI filters can be built directly into the connector body.

Both the 2-mm and the 2.5-mm connectors coexist in IEC 917 standard (which was one reason to adopt a 0.5-mm raster). Proponents of the 2-mm connector boast a higher pin density for their connector. Proponents of the 2.5-mm connectors argue that their connectors can fill the available space without holes and that the connectors can reuse the calculations from the established 1/10th of an inch connectors. The user has to choose. For instance, the developers of the Futurebus opted for the 2-mm raster while Combus designers chose the 2.5-mm raster. The 200 Mbit/s-graded fiber-optic connectors may ultimately replace a large number of pins, easing extraction and insertion. Indeed, with a typical insertion force of 0.7 N/contact, plugging a connector with 255 contacts requires 178 N (corresponding to a weight of 18 kilograms or 36 pounds)!

After years of work, MO is now ready for the market. The IEEE set up the 1301 metric standard group, chaired by Hans Karlsson of Sweden, for computer packaging. DIN and IEC standards have already standardized many parts.[9,10] Siemens's SIPAC and other products are appearing based on these standards. With the exception of the Metral connector, most other components are not yet available in quantities.

How to write metric units

To ease the reading of technical texts, IEEE Standard 268 defines the rules for writing the units and the expressions. Order this leaflet from the IEEE Service Center, 445 Hoes Lane, PO Box 1331, Piscataway, NJ 08855-1331.

Here is a small compendium of some often-ignored rules:

It is important to respect upper- and lowercase letters in the units. For instance, 22 MF means megafarad (quite a huge value), while 22 mF means millifarad (looks more familiar). A crystal with a frequency of 10 mHz (millihertz) is rather unusual (must weigh some kilograms) in contrast to a 10 MHz (megahertz) crystal.

Names of the units always appear in lowercase, even if their unit abbreviation appears in uppercase. For instance, one writes 10 A or 10 amperes.

The standard also defines rules for combining units. Example: A speed of 12 meters per second should be written as 12 m/s, not 12 mps. A torque of 5,0 newton-meters should be written as 5,0 Nm, not 5,0 Nxm.

In expressions, units should be enclosed in square brackets:

Force [N] = Mass [kg] × Acceleration [m/s²]

Std 268 also recommends the symbol of a physical variable. For instance, "I" represents a current, "v" a speed, and "m" a mass. Note that the name of a variable should be different from its unit abbreviation (do not use "V" to express a voltage, but "U").

IEC Directives (part 3) formulate these rules more strictly for all international standards:

- The decimal sign shall be a comma on the line in all languages. Although this rule avoids confusion between the decimal sign and the multiplication sign, it is not prescribed in IEEE Std 268 (For example, write 10,5 and not 10.5.)
- Values shall be separated from the unit by a blank space. (For example, write 10 A and not 10A.)
- Digits shall be grouped in threes, separated by blanks, except for numbers designating years. (For example, 23 456, 2 355 and "in the year 1991" are proper renderings. Apostrophes and commas are banned.)
- In expressions, multiplication is expressed by an "×", not by a point "·". (For example, $1{,}8 \times 10^3$.)
- When using scientific notation, exponents should be a power of 3. (Example: $\varepsilon 0 = 8{,}854 \times 10^{12}$ F/m, not $0{,}885 \times 10^{11}$ F/m.)

In practice, however, the English rules—decimal point, separation of groups by commas, multiplying with a point—are now so widespread in the scientific and engineering field that one cannot ignore them. Word processors, compilers, and calculation programs all use the US schemes.

The MO still has some unsolved problems. Siemens and DuPont cannot agree on the separation between the backplane and the board. DuPont argues that a distance of 10 mm rather than 12.5 mm improves the electrical behavior at high frequency. Indeed, with the short rise time of current and future circuits, the connector inductance cannot be neglected any more.

The major hurdle will be user acceptance: The advantages must be overwhelming for a user to change from an established standard like the 19-inch system. The IEEE P1301 group prepared a transition path: old VME cards and even complete subassemblies fit into the new racks. For instance, the internal width of the cabinet is 483 mm, which readily allows the insertion of a 19-inch (482.6-mm) rack. It's now up to the users to recognize the opportunity.

References

1. IEC 917 1988, "Modular Order," DIN 43 355, IEC Committee, 3, rue de Varambe, Box 131, CH-1211 Geneva, Switzerland.
2. IEC 917-0, "Guide for the User of the Modular Order."
3. IEC 48B, "Connector 2.5 mm," DIN 41 642, DIN 41 650, DIN 43 355, DIN 43 356.
4. IEC 48D (CO) 22, 08/89, Sectional Standard, "Coordination Dimensions for the 25-mm Equipment Practice," DIN E 43 356, parts 1 and 4.
5. IEC Detail Standard 48D (Sec) 11/90, "Specific Dimensions for Cabinets, Racks, Subracks, Plug-In Units, PCBs."
6. IEC 48D (CO) 23, 08/89, "Outside Coordination Dimensions for Cabinets and Racks," DIN 43 356.
7. IEC/ISO Directives Part 3, "Drafting and Presentation of International Standards."
8. W.C. Goeke et. al., "Calibration of an 8 1/2-Digit Multimeter From Only Two External Standards," *Hewlett-Packard Journal*, Apr. 1989, pp. 22-30.
9. A. Goldbacher, "Zentimeter versus Zoll," Part 1, *Elektronik*, July 1991, pp. 64-70.
10. A. Goldbacher, "Zentimeter versus Zoll," Part 2, *Elektronik*, Aug. 1991, pp. 54-61.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 186 Medium 187 High 188

A bountiful return

Carl Warren

McDonnell Douglas

Space Systems Company

(714) 896-3311

x. 6-0669

warren@ssdunx.mdc.com

The last few months have been anything but slow in the world of standards. Those of you interested in teaching systems and software will be happy to learn that IEEE Std P1154/draft 5, Programmed Inquiry Learning or Teaching (PILOT), finally reached the ballot stage on November 12, 1990.

This language specification was developed as a result of experience with an interactive computer system called Computest developed at the University of California at San Francisco in 1962. Computest was used as a method of producing computer-aided instruction systems that allowed the user to peruse a large volume of stored information. The goal of Computest, and now PILOT, was and is to allow the instructor to easily enter information into the system, while providing a robust environment to aid in the teaching process.

The formal development of the PILOT standard began with a project authorization request (PAR) in July of 1987. John A. Starkweather chairs the group. You can contact him at the University of California, Box F-0984, 401 Parnassus Ave., San Francisco, CA 94143; phone (415) 476-7464 or Internet at john@mis.ucsf.edu.

The PILOT standard, which should be a formal IEEE standard by this fall, provides a convenient method of describing instructional material for such computers as the Apple, IBM PC, and Unix systems. One interesting aspect of the standard is that it can be followed to produce a PILOTlike computer-aided instruction course using your favorite language. And Atari fans can enjoy a language such as PILOT, which has some graphics functions.

Big standards at ballot

Those of you keeping tally of certain standards will be interested to know that all of the following standards are at the sponsor ballot stage:

- Futurebus+, Physical Layer and Profile Specifications (P896.2/draft 5.4, January 23, 1991),
- Scalable Coherent Interface (P1596/draft 1.00, January 23, 1991),
- Mechanical Core Specifications for Microcomputers Using IEC 603-2 Connectors (IEEE P1101.1/draft 3.0, November 3, 1990), and
- High-Speed Instrumentation Bus Structure (P1155/draft D2, December 1990).

These are all key standards for future development; the sponsor ballot is one of the major hurdles they have to leap before becoming official standards. The Futurebus+ standard, despite the working group members' patting themselves on the back, appears to have intrigued at least one segment of development for large-scale systems in the US government. Even with some acceptance there, it is unclear whether Futurebus+ can move much beyond that arena. The P896.2 draft for the physical layer and profile specifications, however, may open up a number of new arenas such as desktop (personal computer) applications, and general-purpose I/O systems.

Futurebus+ seems to be aimed more at military applications in which, traditionally, cost and size haven't been considered impediments to development. This tradition may be changing

0272-1732/91/0800-0040$01.00 © 1991 IEEE

though with the new severe restrictions on spending and an approach by military systems designers to downsize for power, cost, and weight.

The standard that may indeed become the hot leader is the Scalable Coherent Interface (SCI). This ballot concerned the logical, physical, and cache coherence specifications. What makes SCI intriguing is that it provides computer-buslike services using a collection of point-to-point links instead of a physical bus to reach higher speeds. This interesting innovation means that processors, workstations, and a wide variety of instruments can all reside in a very large structure that communicates via packet protocols.

This idea isn't new, but the approach taken by the SCI working group is. Moreover, it is in synch with the ANSI X3 committee working on an optical version of the Small Computer Systems Interface (SCSI) called Fiber Channel. I suspect that the SCI standard, especially the definition of cache coherence and the shared distributed-memory, model will most likely influence the development of ANSI's Fiber Channel.

For further information on SCI, you can contact David B. Gustavson, Computation Research Group, Stanford Linear Accelerator Center, PO Box 4349, Bin 88, Stanford, CA 94309; phone (415) 926-2863; fax (415) 961-3530 or (415) 926-3329; or on Internet at dbg@slacvm.slac.stanford.edu. The working group material is available electronically on Internet through him.

The IEEE Standard for Mechanical Core Specifications for Microcomputers Using IEC 603-2 Connectors is part of a series of P1101 specifications for various mechanical cores. The only quibble I have is with the P1101.5 EISA mechanical core. I doubt very much if this has any applicability since EISA is neither standard nor overly functional. Otherwise mechanical core standards (the mechanics of the boards, connectors, board stiffeners, and so on) are necessary to ensure board A from manufacture ZZ will indeed plug into the intended system properly. Moreover, it helps establish the overall real-estate rules for designers to live by. You can obtain more information on this standard by contacting Eike Waltz, IEEE P1101.1 Chair, Schroff, Inc., 1251 Manassero St., Unit 403, Anaheim, CA 92807; phone (714) 777-5795 and fax (714) 777-5984. Apparently, Waltz prefers facsimile communication rather than lengthy phone calls.

Finally, we have the High-Speed Instrumentation Bus Structure, which was just sent for sponsor ballot. This standard establishes specifications for a technically sound modular instrument based on the VMEbus that is open to all manufacturers. The VXIbus is the name for VMEbus extensions for instrumentation. The VXIbus Consortium supports this architecture. This standard may seem a bit tame based on the amount of press VXI has been getting. Yet it appears to be the way to go for high-speed instruments with rapid adoption by manufacturers. The VXIbus appears to be the answer to the age-old question, What good is VME anyhow?

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 180 Medium 181 High 182

Sublime to mundane

A scientific odyssey

The Emperor's New Mind, Roger Penrose (Penguin, New York, 1991, 474 pp., $12.95 paper)

The subtitle of this book is "Concerning Computers, Minds, and the Laws of Physics." It is a far-ranging popular science book along the lines of George Gamow's classic *One, Two, Three ... Infinity*, but it is also united by a common theme: The human mind is more than a Turing machine.

Richard Mateosian

2919 Forest Avenue

Berkeley, CA

94705-1310

(415) 540-7745

You may be thinking that we don't need a book to establish a fact that would be obvious to any child. Penrose agrees to the extent that his title alludes to the fable of the emperor's new clothes. In that story a child points out the obvious fact that the emperor is parading around naked while his adult advisers try to avoid telling him that he has been duped. However, Penrose does not regard the views that he argues against as absurd—he just considers them to be wrong.

His adversaries in this dialog are proponents of artificial intelligence. In particular, the doctrine called strong AI holds that a colossally complicated algorithm can describe the entire functioning of the human mind. Any computing device executing this algorithm would be intelligent, have a mind, and embody all other aspects of what we regard as our consciousness and free will.

Penrose frames this dialog early in the book by citing the writings of John Searle, an opponent of the strong AI position, and Douglas Hofstadter, a supporter of strong AI. Searle imagines himself in a sealed room executing the steps of an algorithm written in English, designed to extract meaning from sentences written in Chinese, a language that he doesn't know. By executing the steps of the algorithm correctly, the imaginary Searle produces the same responses as a Chinese speaker, but he never understands them.

Hofstadter imagines a huge book containing a complete description of the brain of Albert Einstein. By following the instructions in it, he can answer any question exactly as Einstein would have. Thus, Hofstader (or anyone else) using this book would be an instantiation of the algorithm of Einstein's brain. In the strong AI view, any such instantiation would actually "be" Einstein.

Penrose points out flaws in both of these arguments, but more significantly, he sees both sides as accepting the same basic assumption: The effects of any specific physical phenomena can always be accurately modeled by digital calculations. Given that assumption, it is hard to discredit strong AI, and the arguments of people like Searle fall short.

To this point the book rehashes in modern form a philosophical argument that has been going on for hundreds of years. Is the mind something separate and distinct from the physical body? If not, then mustn't its workings be completely determined by physical laws? If so, then how can its workings have any effect on the physical world? These questions constitute the mind-body problem. Penrose takes a novel approach to this problem. Reasoning from Godel's theorem and other considerations, he concludes that human thinking is nonalgorithmic. Reasoning from quantum mechanics, he indicates how this might be physically possible.

To reach his conclusion, Penrose now leads us through computing theory, mathematics, classical physics, relativity, quantum mechanics, and brain structure. Unfortunately, at the end of the journey we realize that his conclusions depend upon a theory of quantum gravitation that hasn't

0272-1732/91/0800-0042$01.00 © 1991 IEEE

been discovered yet. Still, the journey is exciting, and I recommend it to anyone interested in current scientific thought.

Math help

Mathematica Help Stack (Variable Symbols, Berkeley, Calif., $89)

Mathematica is an extremely powerful program for symbolic mathematical calculation, but it is hard to get started using it. Variable Symbols has produced a hypercard stack that provides just what you need. If you have enough memory (5 Mbytes) on your Macintosh, you can run Hypercard and Mathematica simultaneously under Multifinder, so that the help stack is truly available on line. With somewhat less memory, you can run a desk accessory that reads hypercard stacks and still have the on-line help available.

The usefulness of the help stack lies in its organization. There are three ways to access the information you're looking for: the map, the index, and the string search facility. Upon reaching the desired card, you will see that the information is clear and uniformly presented. The official Mathematica documentation appears in a scroll box. Buttons take you to additional cards containing examples, details, or your own notes. The main card for each item contains a syntax example, page references to the official Mathematica book, cross-references to other cards in the stack, and icons that take you to the main divisions of the stack or to the three search mechanisms.

The well-designed stack cards contain enough information but not too much. You can find everything you need, but you aren't overwhelmed with detail while you're still trying to orient yourself. Many hypercard stacks have the appearance of amateur productions. This one seems quite professional, and I recommend it highly.

If you are serious about Mathematica, you should consider a variety of other products and services available from Variable Symbols. The company is independent of Wolfram Research, the supplier of Mathematica, so that it can be objective about Mathematica's strengths and weaknesses. At the same time, Variable Symbols maintains the close ties necessary to do its job effectively. The company's founder, Nancy Blachman, a former employee of Wolfram Research, conducts seminars and Mathematica training courses. I have not attended one, but the notes and my conversations with Blachman convince me they are pretty good.

What does it all mean?

Computer Dictionary (Microsoft Press, Redmond, Wash., 1991, 400 pp., $19.95 paper)

For many years I have been unimpressed by computer dictionaries, but this one is good. Eleven contributors and 10 technical reviewers have provided clear, helpful explanations of many commonly used industry terms. As I paged through the book, I was struck by the clarity and accuracy of the entries. The book was written by professionals who actually use the terms they have included.

When I stopped paging through the book and tried looking up specific items, I was not quite as happy. For example, I tried to find "DOS extender," a term that frequently occurs in books and the trade press. I found "DOS Box" and "DOS Prompt" and a cross-reference to "MS-DOS" (but not "PC-DOS"). Under "extender" I found "extender board." I couldn't find entries for either of the two protocols used by DOS extenders, DPMI and VCPI.

Many years ago I wrote a book called *Programming the Z8000* (now out of print), so I was interested to note that they had an entry for "Z8000." Unfortunately, they call it a descendant of Zilog's Z80. In fact, its architecture and instruction set are completely unrelated to those of the Z80.

Having picked my share of nits, let me reiterate that this is a good and worthwhile reference, and I recommend that you add it to your library. Buy an extra copy and give it to your local high school.

Another trackball

Mouse-Trak (ITAC Systems, Garland, Texas; $179, 9- or 25-pin serial version; $199, bus version)

Because I like the Kensington Turbo-Mouse ADB trackball for my Macintosh (see April and June 1991 columns) I decided to try a trackball for my PC. I have been using the Mouse-Trak for about a week, and I like it just as well as the Turbo-Mouse.

There is an interesting contrast here between the Macintosh and the PC. Every Macintosh comes equipped with a mouse and a standard way of connecting it. Every Macintosh program uses the pointing device the same way. When I received the Kensington trackball, I merely unplugged the mouse, plugged in the trackball, and nothing changed.

The PC's designers did not foresee the degree to which pointing devices would become important. As a result, PC pointing devices must accommodate at least two different connection methods (bus and RS-232) and at least two different message sizes (Microsoft's 3-byte and Mouse Systems' 5-byte). Every PC application must provide its own support for a pointing device. Some do, some don't. Many provide partial support.

The Mouse-Trak provides switches to accommodate these options and driver software to interface with applications. The trackball includes software to allow point-and-click menus to be added to programs (like DOS) that don't support pointing devices.

The Mouse-Trak's nice ergonomic design allows users to work for long periods with CAD programs. The ball rests in a flat surface surrounded by three buttons. A gently inclined padded surface leads up to the flat surface from the front. A fourth small button permits users to toggle between high- and low-precision cursor movements. In the high-precision setting,

the cursor moves one-fourth as much for a given ball movement as in the low-precision setting.

While the Mouse-Trak has four buttons, one of them appears to be unusable with the Microsoft 3-byte protocol. I don't know whether this can be remedied or whether I can use the Mouse Systems 5-byte protocol with my applications. In any event, I like the Mouse-Trak and recommend it for PC use, especially with applications that make extensive and prolonged use of pointing devices.

Wrist rests

Wristsaver (GBM Design, Inglewood, Calif., $18.75 each)
Wrist Pad (Silicon Sports, Menlo Park, Calif., $19.95 each.)

Recently I received samples of two different devices designed to make keyboard use more comfortable. Each sits in front of the keyboard and extends approximately its entire length. They provide a padded place to set your wrists. Since I never learned to type with my wrists off the table, I find they are an improvement over the hard surface. I use one with my Macintosh and the other with my PC, and I can't really tell the difference between them.

Carpal tunnel syndrome is a painful condition of the arms and wrists that results from prolonged repetitive motions (such as those involved in typing. I don't know of any evidence that wrist rests protect against this condition. These manufacturers don't make any such claims, but they don't discourage thinking along those lines. However, I recommend these wrist rests only for the slightly increased comfort.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 183 Medium 184 High 185

Send information for inclusion in Micro News one month before cover date to Managing Editor, IEEE Micro, PO Box 3014, Los Alamitos, CA 90720-1264.

Chip cubes: Can they be mass produced?

Over the last 10 years a small Irvine, California, company has spent $30 million researching ways to stack up to 128 memory or processing chips in a space the size of a sugar cube. Though profits have been small to date, Irvine Sensor Inc. expects its promising Hybrid Mosaic on Stacked Silicon method to catch on with customers by the late 1990s.

The company's high-density approach to 3D chip packaging evolved from the ideas of cofounder and chief scientist John Carson. Supported partially with Pentagon research grants, Carson and cofounder/president James Alexiou helped determine how to make uniform paper-thin layers of silicon and remove from the cube the heat generated by the circuitry. The company secured several patents for new techniques in the process.

Carson reports that the research into such intricate electronics cost three times more than expected. For example, researchers had to look at more than 1,000 different types of glue before they found the right one.

Though it still must find a way to use the chip-stacking technology in mass production, Irvine Sensors has received development contracts from each branch of the armed forces to develop cubes for use in sensor-based missile detection and tracking systems. The company expects these contracts to lead to mass production during the late 1990s.

Future company plans also call for custom manufacturing in low volumes and third-party licensing of the technology for use in high-powered workstations, neural networks, and hard-disk drives for laptop computers. Manufacture of the 3D chips involves 52 steps.

CSMA/CD amendments proposed

ANSI's Accredited Standards X3 Technical Committee X3S3 on data communications invites interested US individuals or organizations to participate in its current liaison project for adding a 10-Mbyte twisted-pair medium to ISO/IEC 8802-3:1989. The development project will result in an amendment to the Information Processing Systems, Local Area Networks Part 3: Carrier Sense Multiple Access with Collision Detection Access Method and Physical Layer Specification.

For further infonmation, contact X3S3 Chair W.F. Emmons, IBM Corporation, PO Box 12195, Dept. C71/673, Research Triangle Park, NC 27709; phone (919) 254-4138; or International Representative John Wheeler, Wintergreen Information Services, Suite 347 Carriage House Commons, 159 West Main Street, Webster, NY 14580; phone (716) 671-4087.

ANSI's X3 committee also announces public review and comment periods from July 12th to September 10th for three New Work Item Proposals. One proposal will add a conformance test for the medium attachment unit to the 10Base5 section of ISO 8802-3, while

0272-1732/91/0800-0044$01.00 © 1991 IEEE

a second will add fiber optics media to the 10Base-F section. The third proposal will update and correct 8802-3 as necessary.

Direct comments and requests for copies of the proposals to Barbara Bennett, X3 Secretariat, 311 First Street NW, Suite 500, Washington, DC 20001-2178; or phone (202) 626-5743.

Accelerating signal-processing research

Defense researchers could save time and money when studying signal-processing techniques for future high-performance computer displays and video-imaging systems. They may now access a "Princeton Engine" video supercomputer created by the David Sarnoff Research Center and located in a new research facility established by the US National Institute of Standards and Technology. NIST officials in Gaithersburg, Maryland, expect the facility to strengthen plans to address the mathematics of signal processing and test and evaluate electronic circuitry.

The US Defense Advanced Research Projects Agency partially supports the research facility.

EIC welcomes back editor

Rejoining the editorial board of *IEEE Micro* after a two-year absence is K.E. Grosspietsch, reports Editor-in-Chief Dante Del Corso. A researcher at GMD (the German national research center for data processing, Gesellschaft fuer Mathematik und Datenverar-beitung), Grosspietsch will review European manuscripts for publication in the magazine.

Grosspietsch received a PhD in computer science from the University of Bonn, is a member of the German computing society, Gesellschaft fuer Informatik, and has served as coguest editor of past issues of *IEEE Micro*.

Current literature

Focusing on one topic each quarter allows *ISA Transactions* to offer readers in-depth articles on current industrial measurement, control, and automation issues. The digest also offers opinion, discussion, and recommendations edited by industry experts.

Instrument Society of America, 67 Alexander Drive, PO Box 12277, Research Triangle Park, NC 27709; (800) 334-6391; $100 (nonmembers), $60 (members).

The FFT: Fundamentals and Concepts discusses Fourier theory with a minimum use of mathematics. Included in the 192-page introductory study are 100 drawings and waveform photographs.

Techniques to implement the SCSI interface can be found in *Fast Track to SCSI: A Product Profile*. Written by Fujitsu Microelectronics, this 208-page manual also informs readers about the company's specific SCSI design features.

Prentice Hall, Book Distribution Center, Route 59 at Brook Hill Drive, West Nyack, NY 10995; $49 (FFT); $24.95 (SCSI).

Micro bits

Beginning this July, the French videotex network, **Minitel,** offered users access to the US-based **Compuserve** on-line information service at fees of US$20/hour (in one-minute increments).

The newly formed **VHDL International, Inc.** provides users of the IEEE Std 1076 hardware description language with up-to-date information and support. It helps in VHDL organization efforts and funds on-going support activities. Potential members should contact Ron Abelmann at (415) 659-0901.

Carnegie Mellon researchers developed a method to reduce the density of defects and increase yields of **electronic materials** grown by organometallic vapor phase epitaxy. They deposit the material on a substrate having a special misorientation or angle. Use of this angle could produce higher quality, smoother layers with improved electronic properties.

Taiwan recently gifted the University of California, San Diego with the only US archive of **74,000 Chinese characters** linked with computer codes in an effort to enhance communication capabilities throughout the Pacific Rim region. Users may access the digitized fonts and codes through local and international computer networks.

IBM continues to adapt to market changes. It announced alliances with **Siemens AG** to produce 16-Mbit DRAMs and with **Borland International** for a new version of OS/2. The company also formed **Integrated Systems Solutions Corp.** from its outsourcing services division, invested $100 million in **Wang Laboratories,** agreed to sell **Lotus Development's** Notes program, and finalized the much-heralded agreement with **Apple Computer** to develop new computers and software.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 198 Medium 199 High 200

Send announcements of new microcomputer and microprocessor products to Managing Editor, IEEE Micro, *PO Box 3014, Los Alamitos, CA 90720-1264.*

Joe Hootman

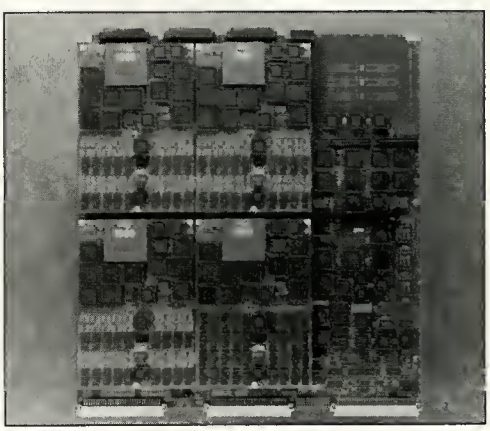
University of North Dakota

Special boards

Gflops multicomputer

Developers of embedded systems can integrate their own system with over 2 Gflops of scalable multiprocessing power using up to eight MC860VS multicomputers. Each MC860VS offers from one to four Intel i860 processors, using modular nodes on a 9U VME card, for up to 320-Mflops performance in one VME slot. A 160-Mbyte/s, configurable Interboard bus provides i860 communication between MC860VS boards. The MC/OS Version 2 software provides a real-time i860 kernel for simulation, signal-processing, and image-processing applications. The multicomputer supports Sun-3/-4 and Silicon Graphics workstations as well as 680X0 and Sparc-based single-board computers with Unix or Vx Works. *Mercury Computer Systems; from $24,300.*

Reader Service No. 10



Mercury Computer Systems MC860VS

i860 real-time color graphics board

Superstation 3D is an integrated, real-time, color graphics, arithmetic accelerator display board for ISA/EISA-based personal computers. Targeted at CAD, 3D-rendering, animation, multimedia, and scientific-visualization applications, the TMS34020/i860-based board accelerates computation and reduces dependence on the host CPU.

With 2 to 16 Mbytes of program memory and genlock capability, Superstation 3D supports the synchronization of RGB output to international broadcast video standards. *Hercules Computer Technology; from $4,495 to $5,895, depending upon memory configuration.*

Reader Service No. 11

Cut PCB development time

Designed for robotics and CPU-intensive process control applications, the CP-332 single-board computer features the Motorola 68332 microcontroller and an optional 68881 floating-point unit.

The 5.5 × 8.5-inch board comes either fully assembled and tested with 256 Kbytes of EPROM and 256 Kbytes of RAM or as an unpopulated bare board with documents. *Allen Systems; $750 or $125 (bare board).*

Reader Service No. 12

Graphics controller

The Gesvig-24 CAE, simulation, and imaging controller draws 20 million pixels per second with a 32-bit Hitachi graphics drawing processor. Capable of displaying color pictures with 1,280 × 1,024 pixels on a 110-MHz monitor, the 100 × 160-mm, single-height Eurocard module contains 2 Mbytes of dual-port video RAM and an 8-bit color lookup table.

A GDP chip or host processor can access the Gesvig-24's video memory simultaneously as can the bus in a full-bitmap configuration. *Gespac; $2,950.*

Reader Service No. 13

0272-1732/91/0800-0046$01.00 © 1991 IEEE

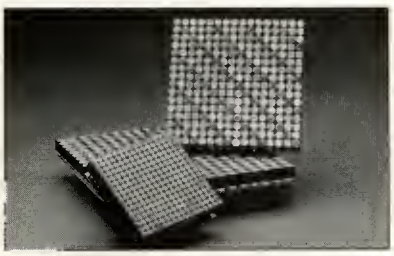
Display devices

Graphics display tiles

Users of LU256X square display modules can control individual LEDs in real time with a personal computer as part of a multitasking environment. The 16 × 16 dot matrix outputs luminous, switchable red, green, or yellow displays in each of its 256 pixels.

Intended for use in large graphic panels such as athletic scoreboards and moving message displays, LU2563-5MU subsystems come in 64 × 64-mm, 96 × 96-mm, and 144 × 144-mm sizes. *Rohm Corp.; $100 per module (1,000s).*

Reader Service No. 14



Rohm Corporation LU256X series

EL display, computer

Users of portable instrumentation and machine embedded-control applications may want to look at this electroluminescent monitor and flat-panel computer for crisper displays.

According to the company, the EL4836LP display provides a 2-to-1 increase in brightness and a 17-to-1 improvement in contrast ratio over supertwist, backlit LCD versions. The display offers a 160-degree viewing angle and requires 1.3 watts of power.

The 3-inch-deep Planar/AT computer with flat-panel display supports factory and machine embedded control applications with a 16-MHz 80286 and 640 × 400-pixel CGA/EGA touch screen for easy use. *Planar Systems.*

Reader Service No. 15

Plotter replaces 57000 series

The 67436 electrostatic plotter draws three times faster with higher throughput, twice the number of fill patterns, and eight times the number of line types than the earlier 57000 series. Wide-format (36-inch) monochrome plots come directly from the company's CCRF data or CCGL random vector data in 400-dpi resolution. Users may define and specify 512 fill patterns, 8,192 colors, and 8,192 line types with the four input-port plotter. *Cal Comp; $32,900.*

Reader Service No.16

Cal Comp plotter

Four brightness levels

Model 3601-97-032, the Ninety Series Flip vacuum fluorescent display module, supports large-volume OEM applications such as point-of-sale terminals, medical equipment, and telecommunications. The intelligent module with microprocessor controller provides scan, refresh, and data I/O operations, displaying two lines of 16 characters in a 5 × 7 dot matrix. *Industrial Products Division of IEE; $88 each (100s); 4-6 weeks ARO.*

Reader Service No. 17

Communications

LAN adapters link PCs/laptops

A series of external LAN adapters link a personal computer or laptop to three cabling systems: thin Ethernet, 10Base-T, and Arcnet. Model LAN-10BT connects to the 10Base-T systems using twisted-pair wires and standard RJ45 phone jacks. Model LAN-CX connects to thin Ethernet and 10Base-2 systems with a BNC coaxial connector. Model LAN-ARC connects to Arcnet RG-62 networks with a BNC connector. Inherent driver software supports Novell Advanced Netware 286 and 386, 3COM 3+Share, Net Bios, TCP/IP, and IBM PC LAN and PC-NFS networks. *Solectek Corporation; $499.95 and $399.95 (LAN-ARC).*

Reader Service No. 18

Printer extenders

Locating a parallel (Centronics) printer up to 2,000 feet from a computer is possible with the Model 264 printer extender. A transformer isolation on one twisted-pair wire protects the computer and printer from transient overvoltages and surges. Model 264 contains one unit that connects to a computer's parallel port via a DB-25 male connector and another unit that connects to the printer port via a standard 36-pin male connector. *Telebyte Technology; $115 each.*

Reader Service No. 19

CATV modems

The C-series data modems—C10, C20, and C60—work with existing community antenna television plants as their data transmission medium. Designers can use the 9,600-bps C-10 for simple telemetry and data applications, the 19,200-bps C20 for synchronous and asynchronous data communications, and the 64,000-bps C60 for synchronous applications. The latter two modems feature programmable set-up via a hand-held programmer. *C-COR Electronics.*

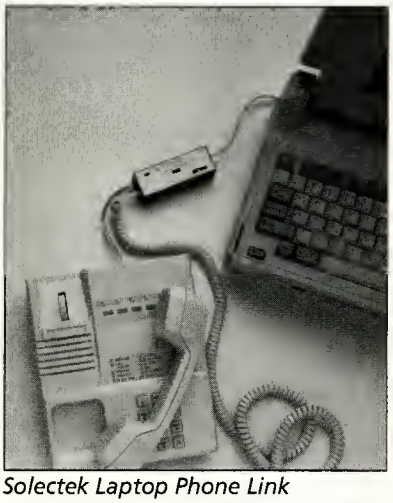
Reader Service No. 20

Laptop-to-phone linkage

The Laptop Phone Link connects a modem or fax directly to phone systems without a dedicated line. The stand-alone unit attaches to the standard RJ11 connection on the phone handset and requires no software or external power. The Laptop Phone Link operates with modem-equipped laptop

or desktop computers and fax machines. *Solectek Corp.; $119.95; $149.95 (for modems and faxes requiring line voltage).*

Reader Service No. 21



Solectek Laptop Phone Link

Computer systems

Real-time RISC

Designed for performance monitoring and control graphics application requirements, the Model 7100 RISC workstation offers a real-time kernel that uses the AT&T Unix Version 5.3.2, transmission-control and Internet protocols for Ethernet, and VME/VSB bus expansion slots. The 7100's graphics capabilities include 1,280 × 1,024 display resolution, 12 display planes, and high-resolution color CRT monitors. Available in tower or rack enclosures, the basic system features a 19-inch color CRT, a 90-Mbyte hard disk, a 150-Mbyte tape drive, and software. *Aydin Controls; $35,100.*

Reader Service No. 22

Notebook power

A 486 microprocessor provides the foundation for Notebook Computer models. The 486-T20 offers a 20-MHz 486SX microprocessor with a math coprocessor emulator and a 20-Mbyte hard disk. The 486T33 features a 33-MHz 486DX microprocessor with either a 40- or 60-Mbyte hard disk. The computers weigh 4.5 pounds, including the battery, and measure 8.5 × 11 × 1.4 inches. *Notebook Computer Co.; $4,495 (486-T20) and from $6,795 (486-T33).*

Reader Service No. 23

VMEbus computer

The XVME-684 two-board PC/AT computer offers a 25- or 33-MHz 486 processor, 4 or 16 Mbytes of dual-access DRAM, and both VMEbus and PC/AT hardware interfaces. It features an IDE hard disk and floppy disk controllers, a Super VGA graphics controller, and two RS-232C serial ports. A complete VME interrupter and interrupt handler and software programmable-byte-swapping logic are also included. *Xycom; from $6,700.*

Reader Service No. 24

RISC-to-bus connector

The Model 446 adapter, combined with the Model 400-921 support software, connects a RISC System/6000 computer to a VMEbus system. Designed for applications that do not require full RISC System/6000 capabilities, the adapter allows an AIX application to operate as a single-board bus master processor on the VMEbus. The software, while not required for the adapter, eases the development of application-specific drivers for the RISC System/6000. *Bit 3 Computer Corp.; $1,995 each (Model 446), $600 per license (Model 400-921).*

Reader Service No. 25

C, C++, and Unix products

C reference guide

A 72-page pocket-size reference book for the Unix System V, Release 4, summarizes C library functions and system calls. The *C Library Reference for Unix System Release 4 (with ANSI C)* guide presents new system calls for virtual file systems, networks, multiple processors, and the programmable process scheduler. The authors also flagged standard ANSI library functions to assist programmers in writing portable code. *Specialized Systems Consultants; $8 each.*

Reader Service No. 26

Unix tools with source code

Nine applications and utility software packages that work with most Unix machines contain complete source code and documentation to allow for program customization and portability. Tool packages for programming, engineering, system administration, finances, and multimedia are available. Users can purchase site licenses, floating licenses, and binary versions of the programs. *Full Source Software Corp.; $149 each.*

Reader Service No. 27

Object-oriented testing

The Test Coverage Analysis Tool for the C++ Language extends the present TCAT system via a new instrumenter system and the Cover++ post-test coverage analyzer. Designed for object-oriented applications, TCAT/C++ uses a C1++ coverage metric that measures aggregate and individual object type for each C++ method. The analysis tool supports the following platforms: IBM RISC System/6000, Sun/Sparc, DECstation, HP/9000, and most Unix- and MS-DOS-compatible PCs. *Software Research; $1,900 (single-user license) to $21,600 (workstation license).*

Reader Service No. 28

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 189 Medium 190 High 191

Product Summary

Joe Hootman
University of North Dakota

Manufacturer	Model	Comments	R.S.#
Computer systems			
Park Engineering Associates	Comp Cap	Weighing 1 pound, this computer comes in two versions: one built directly into a hardhat and a head-band model with the electronics built into a belt or vest. Its Private Eye 12-inch monitor weighs 2 ounces and displays 720 × 280 pixels. The computer also includes 4 Mbytes of on-board RAM and a voice recognition system. From $1,500, 30 days ARO.	80
Sanyo Icon	88open series computers	Based on Motorola's 88100 32-bit RISC chipset, the systems can handle 18 to 250 user terminals in tower or rack-mount configurations. Each system has a maximum capacity of 128 Mbytes of on-board main memory and 44.4 Gbytes of unformatted disk space using SCSI or HSMD disk drives. Internal installation of industry-standard half-inch tapes and support of high-density cartridge and cassette tape back-up are other features. From $38,514.	81
Software tools			
Applied Microsystems Corp.	20-MHz MC68302 development tools	The tool set supports the real-time performance of the MC68302 with an ES-1800 emulator that allows DMA operations to run whether the microprocessor is active or paused. The ES-1800 equipped with an SCSI permits high-speed communications with the host computer. Users can choose between symbolic or high-level-language debugging environments. From $18,270 (20 MHz); $3,500 for upgrading an existing 16-MHz MC68302.	82
Multilingual Software	Multiwriter	A PC word processing program capable of working in 30 languages, Multiwriter also offers users bilingual versions, including Russian, Polish, Hebrew, and Slavic Writer. It supports Hewlett-Packard-compatible laser printers and 9- and 24-pin dot-matrix printers. $199 (Multiwriter); from $99 (bilingual versions).	83
Software Interphase	Lite Switch	Upon switching from one program to another, this utility program saves the original program to disk or EMS, which frees computer resources for the next program. Using less than 7K of memory, the utility switches programs in under one second and works in standard text and graphics modes. $59 (single-user version), $99 (network version).	84
Miscellany			
Burr-Brown Corp.	DF1700	Accepting 16-bit input data, the dual-channel CMOS digital filter also permits the selection of 16-, 18-, or 20-bit output data. It contains dual filters that each consist of three cascaded, 2X oversampling finite impulse response filters. The DF1700 serial output is compatible with selected Burr-Brown audio digital-analog converters. $14.90 (100s).	85

continued from p. 19

ule so that contention does not occur on any Y port at runtime.

Reconfigurability of interconnection topology. According to the operation modes of the crossbar LSI module, the entire crossbar network also provides the following three operation modes:

- **Monolithic demand.** All 256 crossbar LSI modules and 128 PEs operate in the demand mode. Contentions between connection requests should be arbitrated on demand at runtime. Arbitration occurs at two levels as shown in Figure 3: local arbitration in an LSI module with respect to a Y port, and global arbitration by a message receiver with respect to the associated column bus. This mode works best in applications with irregular data structures and communication patterns that cannot be analyzed at compile time.
- **Monolithic preset**. All 256 crossbar LSI modules and 128 PEs operate in the preset mode. Arbitration should be completed prior to program execution (at compile time) to prevent any contention from occurring at runtime. Each message receiver, along with 16 Y ports on the corresponding column bus, can easily establish the interconnection path according to the current switching pattern. It is possible to alternate the switching patterns in time-sliced, round-robin fashion.

  As shown in Figure 4, a tree structure and a neural network can be simulated by both of the 3 switching patterns, respectively. This mode seems to best suit limited applications with deterministic communication patterns, such as partial differential equations with a regular 3D, nearest neighbor-mesh topology; quantum chromodynamics with a regular 4D, nearest neighbor-mesh topology; and fast Fourier transformations with a regular hypercube/omega topology.
- **Partitioned.** In this mode 256 crossbar LSI modules and 128 PEs are partitioned into up to 16 groups, each of which independently operates in either the demand or the preset mode. Each group consists of $8 \times n$ PEs and n^2 LSI modules, where n is in the range 1 to 15. Figure 5 on page 52 demonstrates the case of two clusters work-

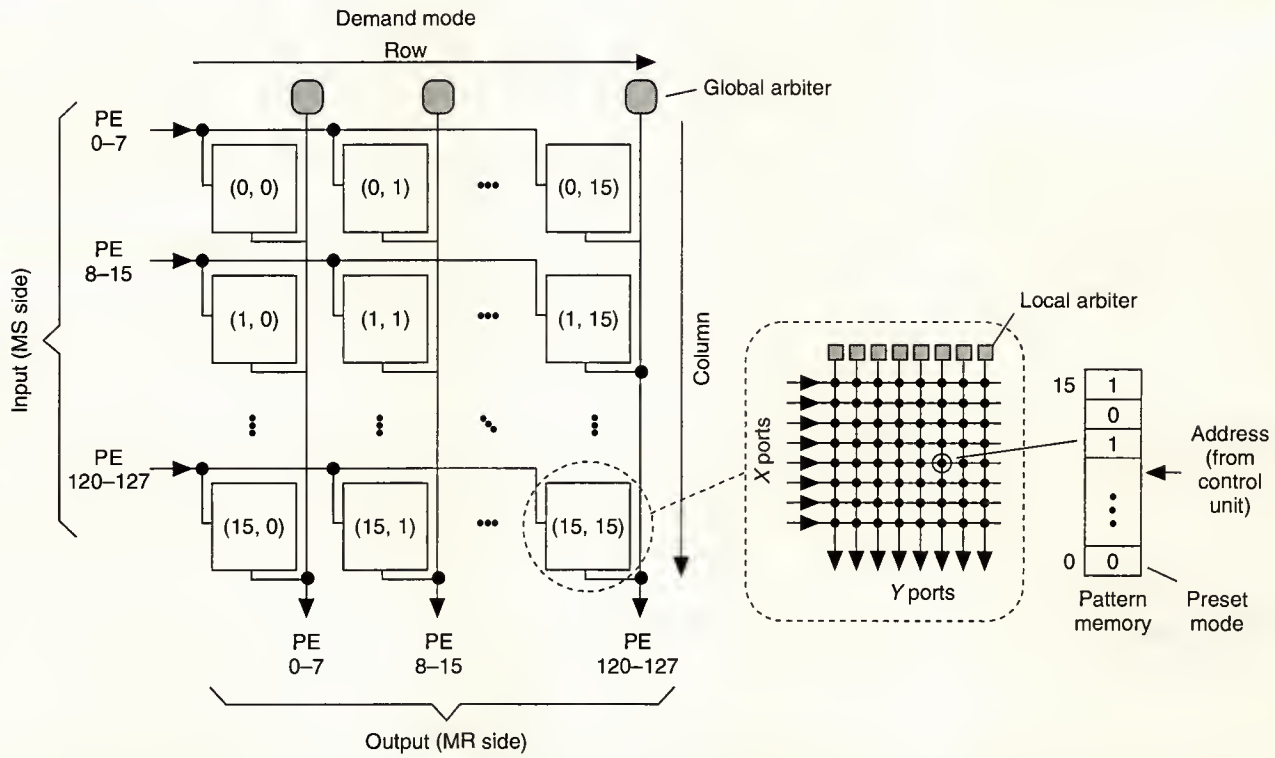


Figure 3. Configuration of the 128 × 128 crossbar network.

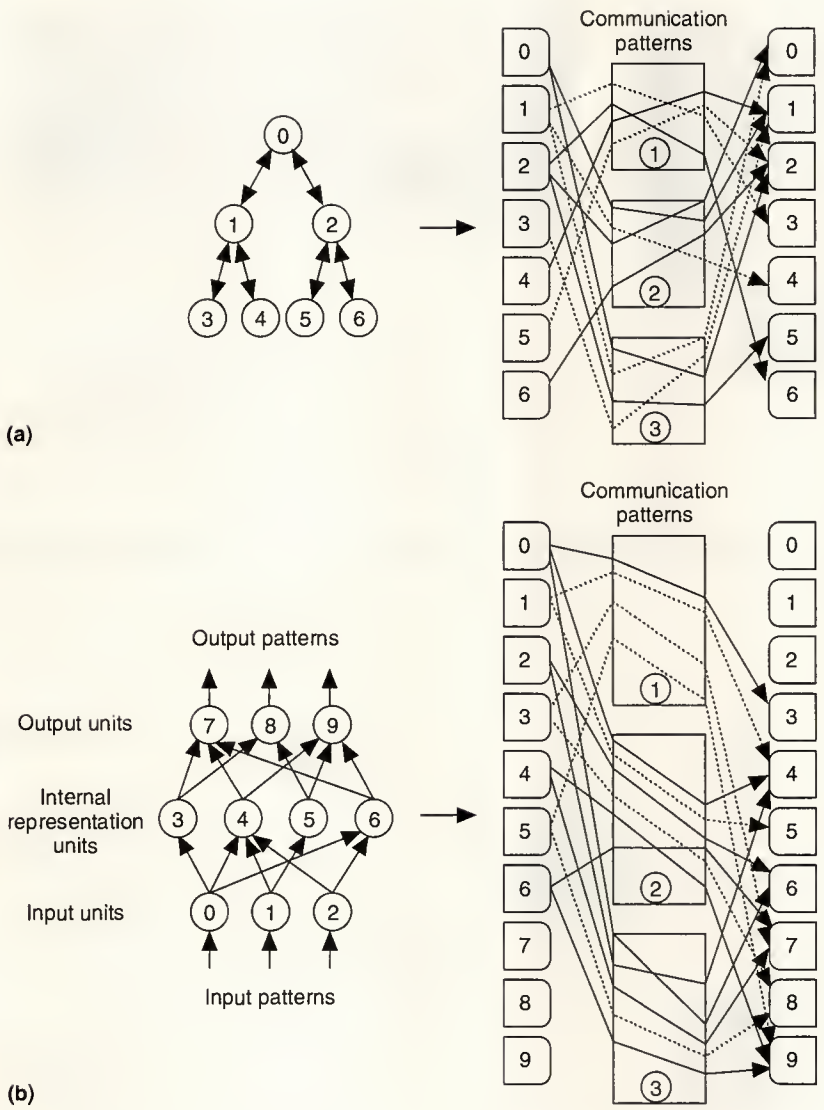


Figure 4. Mapping topologies to the crossbar network by using the preset mode: binary tree (a) and neural network (b).

ing cooperatively. Users can process images on a mesh topology and use the results in an artificial intelligence application on a tree topology.

Memory architecture. As shown in Figure 1, the KRPP employs a distributed-memory organization and provides no global memory. To exploit the memory reconfigurability however, the KRPP allows more than one processor to share a local memory. The processor in each PE can directly access the local memories (called remote memories) of the other PEs via the 128 × 128 crossbar network. The memory architecture is referred to as local/remote architecture. As a result, the KRPP encompasses a shared-memory tightly coupled multiprocessor and a message-passing loosely coupled multiprocessor.

Some multiprocessor systems such as the CMU Cm*,[10] the IBM RP3,[11] and the BBN Butterfly[12] use local/remote architectures. These architectures, including ours, differ in their addressing schemes.

Figure 6 schematically depicts the addressing scheme of the KRPP, which is achieved by two-level address translation. The scheme requires three address representations: virtual, real, and physical. The two-level address translation occurs as follows.

First-level mapping (first-level address translation). For PE_m the memory management hardware maps a virtual-address space VAS_m of 4 Gbytes to the real-address space RAS_m of 4 Gbytes. The memory management unit MMU_m translates a 32-bit virtual address VA_m generated by a Sparc microprocessor into a 32-bit real address RA_m with conventional two-level paging. The page size is 4 Kbytes.

Partitioning. The hardware partitions each real-address space into private and common spaces:

- **Private.** For PE_m a low-order, 2-Gbyte space (the most significant bit of the real address is 0) is its private space PS_m, which is identical to its physical-address space PAS_m. Thus a private-space real address $PSRA_m$, excluding its most significant bit, is directly used as physical address PA_m.
- **Common.** A high-order, 2-Gbyte space (the most significant bit of the real address is 1) is common to all PEs. The common space comprises 128 shared-memory windows, each of which is 16 Mbytes in size and segmented into 4K window pages. The hardware assigns a window to a PE that is indexed by the PE number.

Second-level mapping (second-level address translation). When PE_m targets a remote memory reference of a shared-

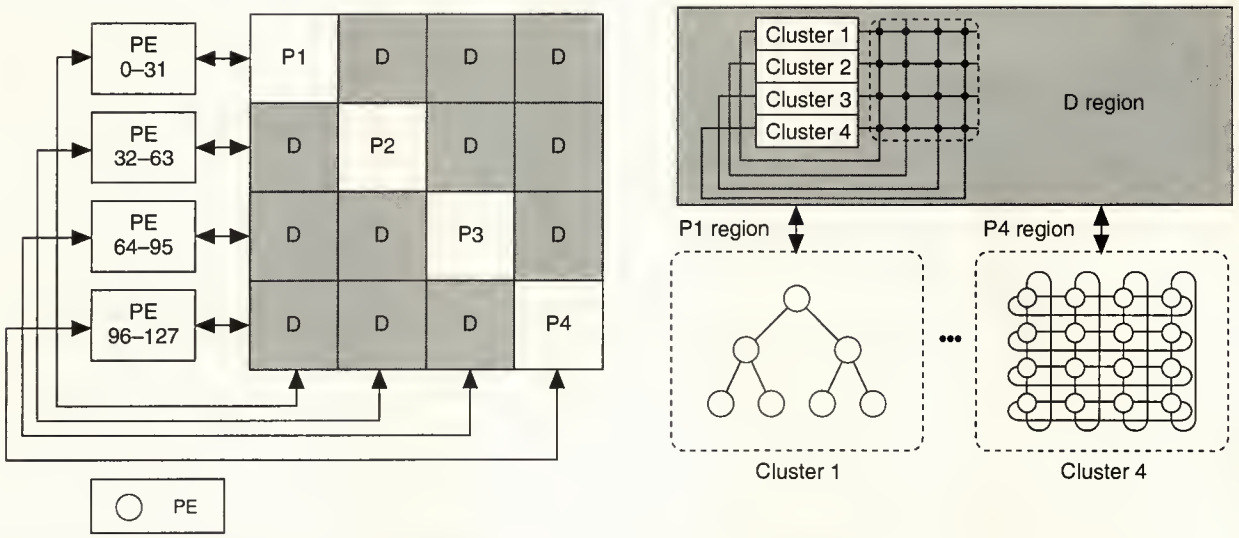


Figure 5. Clustering in partitioned mode.

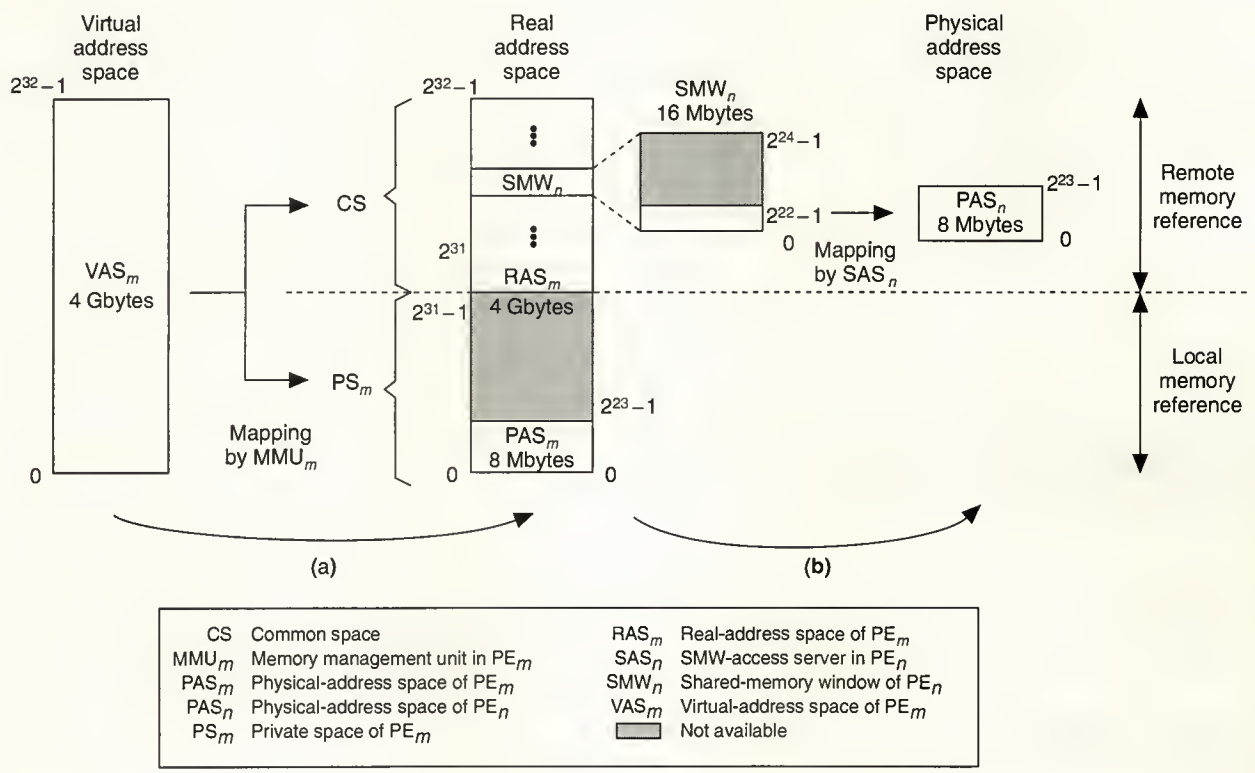


Figure 6. Addressing scheme: mapping VAS_m to RAS_m (a) and mapping SMW_n to PAS_n (b).

memory window SMW_m, the KRPP forwards an SMW access request from PE_m to PE_n via the crossbar network in message form. On PE_n the hardware maps the target SMW_n to its physical-address space PAS_n. That is, the shared-memory-window access server SAS_n translates the received common-space real-address CSRA, excluding a portion of shared-memory-window index, into physical address PA_n with conventional one-level paging.

From a processor or local memory viewpoint, the addressing scheme does not differentiate between local and remote memory references. Both references perform memory reads and writes equally for all data sizes (byte, half word, word, and double word). The only difference appears in the local and remote memory access time ratio.

This addressing scheme allows us to configure the KRPP as either a shared-memory tightly coupled multiprocessor, a message-passing loosely coupled multiprocessor, or a hybrid of the two. Mapping all of the virtual-address spaces to the common space (the private space) results in the system becoming a shared-memory tightly coupled multiprocessor (a message-passing loosely coupled multiprocessor). In addition to this, setting a hardware switch lets us change the memory architecture of the system to either of two types of shared-memory tightly coupled multiprocessors: uniform-memory access and nonuniform-memory access.

The KRPP also provides each PE with a private cache. To maintain cache coherence, we take software- and hardware-level approaches. The hardware-level approach includes a limited directory scheme that permits only one copy.[4]

Parallel operating system

An operating system should reflect the flexibility and reconfigurability provided by the KRPP. However, we take a step-by-step approach in building an operating system. The first step involves building an operating system for the shared-memory tightly coupled multiprocessor, for two reasons. 1) This system configuration promises potentially high performance. We believe that exploiting an operating system that can extract the potential gain provided by this type of system contributes to research on highly parallel processing including operating systems. 2) Exploiting such an operating system would also give us some quantitative and qualitative information on the effective use of the flexibility and reconfigurability provided by the KRPP.

We adhered to the following design principles for the operating system:

- **A well-matched parallel processing model.** Providing users with this model to the shared-memory tightly coupled multiprocessor is an important consideration. The operating system provides a process-thread model in which a process consists of an execution environment and multiple control flows (threads). The execution environment includes a paged virtual-address space and protected access to system resources. A thread is the basic unit of CPU utilization (a control flow).[13] It contains a context: a program counter and the contents of registers. We are sure that the process-thread model is well matched to this type of architecture, because all threads in a process share its virtual address space. The threads in a process work cooperatively. Therefore, considering the model, we chose scheduling and memory management schemes.
- **Exploiting parallelism in a kernel.** To provide a high-performance kernel, we employ a kernel processing style in which: 1) The code for executing the system call is partitioned into various functions, which execute concurrently or in parallel, if possible. We call this first-class parallelization. 2) This style also lets a system call process in multiple control flows, each of which has the same code but different data from each other, if possible. The control flows execute concurrently or in parallel, depending on the type of system calls. We call this second-class parallelization.

We implemented a scheme that realizes both the first- and second-classes of parallelization by using a message-oriented paradigm in the kernel, as shown in Figure 7. When the kernel on a processor (MPU 0 in Figure 7) receives work that can be parallelized, it places request messages into the corresponding queues for idle processors to handle the requests in parallel. When the kernel receives first-class parallelization work, the kernel partitions it into subworks, each of which has different functions. The kernel places messages for the subworks into the corresponding queues.

In second-class parallelization, the kernel places the message or messages into the corresponding single queue. A

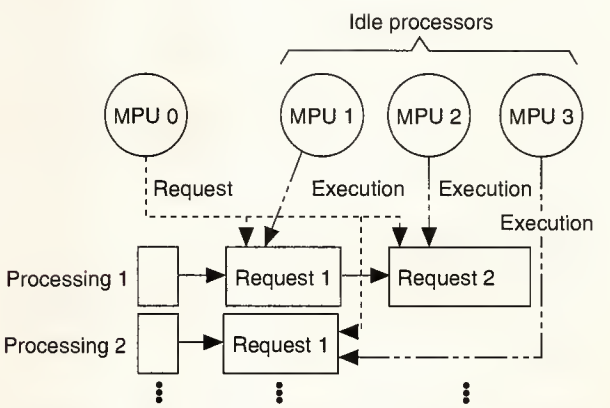


Figure 7. Parallel processing in the kernel with the message-oriented paradigm.

typical example of second-class parallelization is the case with a multiple-thread-creation system call issued, as described later in detail. The scheme allows multiple threads to be created quickly and in parallel. Performance of the scheme depends on trade-offs between enhanced performance due to parallel processing and overhead due to message construction and memory contentions. We pursue the viability of this scheme.

Two-level scheduling. Though a scheduling policy significantly determines system performance, scheduling policies for general-purpose, multiprogrammed multiprocessors are not well understood. Tucker and Gupta[14] and Black[15] have proposed scheduling policies for multiprocessors.

We based our scheduling scheme on a multilevel feedback-priority mechanism implemented in two levels. The first level (process scheduling) allocates processes rather than threads to the processors. A priority is associated with a process rather than a thread, because if a priority is associated with a thread, threads with different priorities in the same process may not run at the same time. A process scheduler determines the process to run with a priority-scheduling algorithm. That is, the activated process scheduler on a processor examines a process ready queue and assigns the processor to a process with higher priority.

The second level (thread scheduling) schedules threads in a process to processors on a first-come, first-served basis. Processors allocated to a process by a process scheduler activate the thread schedulers. They examine the corresponding process-associated thread-ready queue and execute the threads until termination or the preemption of another process with higher priority. When a thread scheduler finds no further threads in the queue, the scheduler activates the process scheduler.

Two-level scheduling tends to allow a processor allocated to a process to continue executing runnable threads in the process. Therefore, the number of address-space switching can be reduced. In addition, both schedulers are separately configured, which provides a wide degree of flexibility in implementing various process allocation and thread-dispatching policies. Future implementation of the thread-dispatching policies includes that of the dynamic thread-merging/partitioning policy. This policy allows threads in a process to be merged dynamically or partitioned according to available processors.

Multiple-thread creation. The operating system provides a multiple-thread-creation system call, and the kernel has a thread control block associated with a thread. This block contains the context of a corresponding thread. To create a thread, the kernel must:

1) examine a list of free thread control blocks and pick up a free block,
2) set the appropriate information in the block,
3) allocate a stack for the thread, and
4) send the block into the corresponding thread-ready queue.

In sequentially processing a system call, the kernel uses considerable execution time whenever a great number of threads must be created. Even with idle processors, only one processor would handle the system call. To reduce the processing time, we exploit parallelism at this level. The parallel processing scheme allows multiple threads to be created in parallel by idle processors.

An overview of the current implementation follows:

1) The kernel accepts the system call to enqueue a thread-creation request message for the dedicated queue. The request message contains the number of threads to be created.
2) Each of the idle processors examines the queue and creates an appropriate number of threads. Processing creation of a thread consists of the previous four steps.
3) The processors continue to handle the request until all threads are created.

Performance of the scheme depends on trade-offs between enhanced performance due to parallel processing and overhead due to message construction and memory contentions. The simulation results show that parallel thread creation takes place 10 to 20 times faster than in sequential operation.

Future plans. We plan to build the following mechanisms into the future version of the operating systems using the reconfigurability of hardware architecture:

- **On-demand reconfiguration of network topology.** Consider a case in which parallel processing consists of two time phases. In the first time phase tasks communicate nearly at random with each other, while in the second, tasks communicate according to fixed communication patterns. A neural network is an example of this style. The first time phase operates in the monolithic demand mode, and the second occurs in the monolithic preset mode in which the fixed communication patterns are set.
- **Processor migration**. In the monolithic preset mode, when tasks on a processor frequently communicate with tasks on other distant processors, the preset patterns are reset. That is, the processor migrates rather than the tasks. This approach results in fast communications between tasks.
- **On-demand reconfiguration of a memory paradigm.** Consider a case in which one task accesses data after multiple tasks access the data mapped to the common space of the real-address space. When the single task accesses the data, the hardware remaps the data to the

private space, enabling the single task to quickly access the data.

Superscalar processor

Superscalar processors improve uniprocessor performance beyond the level of RISC performance by exploiting more instruction-level parallelism.[16] In 1987 we proposed a superscalar architecture called SIMP (single-instruction stream, multiple-instruction pipelining).[5] SIMP could be scalable and object-code compatible with respect to the degree of superscalar operations (the number of instructions issued per cycle). The SIMP architecture implicitly uses dynamic code scheduling[17] to keep sustained performance close to peak performance.

Although the SIMP prototype had tried to alleviate the Flynn bottleneck,[6] it also introduced new bottlenecks. Each bottleneck came from the dynamic code-scheduling algorithm we developed. Simulation results show that SIMP sacrifices sustained performance for scalability, object-code compatibility, and precise interrupts. Since then, we have made new architectural trade-offs between hardware and software and proposed a new superscalar processor architecture, which we call DSNS (dynamically hazard-resolved, statically code-scheduled, nonuniform superscalar). To evaluate the viability of the DSNS architecture, we have been developing a DSNS processor prototype.

DSNS architecture. The DSNS name evolved from our use of the following major design alternatives as attributes for classifying superscalar architectures.

- **When are hazards resolved?** The three classes of hazards (or dependencies), structural, data, and control, can be detected and resolved either statically (at compile time) or dynamically (at runtime). When hazards are statically resolved, superscalar processors have no pipeline-interlock hardware and rely on software pipeline-scheduling techniques to eliminate all the hazards at compile time. On the other hand, when hazards are dynamically resolved, superscalar processors provide hardware pipeline-interlock logic that detects a hazard and stalls the pipeline until the hazard is cleared.

  Although the absence of pipeline-interlock hardware makes the processors simple and fast, the statically hazard-resolved superscalar processors suffer under an increased code size from the insertion of no-op instructions. The problem can be more severe when a multicycle-operation instruction causes a data hazard and when the instruction uses a nonpipelined functional unit, causing a structural hazard.
- **When are the instructions scheduled?** A system can exploit instruction-level parallelism by scheduling the instructions around structural and data hazards, either statically (at compile time) or dynamically (at runtime).

Superscalar processors improve uniprocessor performance beyond the level of RISC performance.

When codes are dynamically scheduled, superscalar processors rearrange the sequence of instruction execution according to the runtime availability of resources and operands. Thus the instructions can begin and complete operations out of order with respect to the compiled code sequence. Two well-known techniques for dynamic code scheduling[18] are Thornton's scoreboarding[19] and Tomasulo's register renaming.[20] On the other hand, when codes are statically scheduled, superscalar processors contain no runtime-scheduling hardware, and the instructions begin operations in order with respect to the compiled code sequence.

Dynamically code-scheduled superscalar processors have at least one advantage over statically code-scheduled processors: They can handle hazards that are unknown at compile time, for example, data dependences due to memory references. However, the additional hardware necessary to implement dynamic code scheduling is complex and expensive.
- **Do instruction-class conflicts occur?** The superscalar processor of degree n can be organized uniformly and nonuniformly.[21] Uniform organizations duplicate all functional units n times, including the instruction fetch and decode logic, register ports, bypasses, and buses. Nonuniform organizations duplicate only the instruction fetch and decode logic, register ports, bypasses, and buses. If necessary, they duplicate some (not all) functional units.

  In nonuniform superscalar processors, since all the functional units are not duplicated, resource conflicts (structural hazards) keep some combination of instructions from being accommodated. Thus instruction-class conflicts arise. Uniform superscalar processors do not incur these class conflicts, but their performance cannot be cost-effective.

Superscalar taxonomy. Statically hazard-resolved superscalar processors are substantially statically code-scheduled. Dynamically hazard-resolved superscalar processors, on the other hand, have two options: static or dynamic code scheduling. Thus superscalar architectures fall under three primary categories:

DSNS must check for structural and data hazards when it tries to issue new instructions.

- SS (statically hazard-resolved, statically code-scheduled),
- DS (dynamically hazard-resolved, statically code-scheduled), and
- DD (dynamically hazard-resolved, dynamically code-scheduled).

The categories are orthogonal to whether they are uniform or nonuniform, and finally we get six categories of superscalar architectures: SSU, SSN, DSU, DSN, DDU, and DDN.

Accordingly, statically scheduled and dynamically scheduled superscalar processors in Smith et al. fall into the SS and DD categories, while the Stanford University Torch and Match fit into SSN and DDN superscalar processor classes.[22] The Intel 80960CA and IBM RS/6000, which are commercial superscalar processors, fall under DSN and DDN classifications respectively. Very long instruction-word machines are an outgrowth of SSN superscalar processors.[23,24] SIMP is the only DDU (and the only uniform) superscalar architecture currently proposed. DSNS is literally a DSN superscalar architecture.

Architectural trade-offs. We based the design of DSNS on three new architectural trade-offs between software (compilers and operating systems) and hardware.

1) The DSNS performance could not be scalable with respect to the degree of superscalar operations. Static code scheduling, instead of dynamic code scheduling, affects performance. Unless a program is rescheduled for a particular degree of superscalar operations, the object code may not be executed effectively on a superscalar processor of that degree.
2) DSNS should be object-code compatible with respect to the degree of superscalar operations; hardware remains responsible for the object-code compatibility. Even if a program is not recompiled for a particular degree of superscalar operations, the object code can be executed on a superscalar processor of that degree.
3) DSNS relaxes the conditions for implementing precise interrupts, and therefore interrupts can be imprecise. Even programs that cause imprecise interrupts, however, can be restarted with the help of operating systems. Thus a new interrupt contract between hardware and the operating systems should be defined.

DSNS processor prototype. The major architectural features of this prototype include 1) dynamic hazard detection and resolution, 2) branches, and 3) a dual-register file.

Dynamic detection and resolution. Since DSNS is a dynamically hazard-resolved, nonuniform superscalar processor, it must check for structural and data hazards when it tries to issue new instructions. Structural hazards can arise from resource conflicts for functional units, register-file read ports, and register-file write ports. Before instruction issue, the system checks only for conflicts on functional units and register-file read ports; it checks write-port conflicts after the issue.

Combinations of instructions that can cause structural hazards depend on the data path configuration, while data hazards can arise from storage conflicts for registers and memory. The system checks for read-after-write, or flow dependence, and write-after-write, or output dependence, hazards due to register conflicts, when new instructions issue. Since all instructions read their registers at the same time, no write-after-read, or antidependence, hazards will occur. The system uses register scoreboarding (not Thornton's scoreboarding) to detect read-after-write and write-after-write hazards.

When a structural or data hazard exists, the system stops issuing the instruction and holds it until the hazard is cleared.

Detecting and resolving data hazards due to memory conflicts is not as simple as dealing with resource or register conflicts. The DSNS processor prototype provides two identical pipelined load/store units and a nonblocking data cache with two ports. Thus it can access caches out of order. Three possible data hazards exist: load-after-store, or flow dependence; store-after-load, or antidependence; and store-after-store, or output dependence. Detecting and resolving these data hazards at runtime means implementing a number of address comparators and interlock hardware in the load/store pipelines. The DSNS architecture, however, does not assume such hardware; instead, it imposes memory disambiguation on compilers.

To make this scheme tractable, the DSNS architecture defines three types of load/store instructions:

- A strongly_ordered load/store instruction accesses the cache in order with respect to the compiled sequence of any load/store instructions.
- A weakly_ordered integer (or floating-point) load/store instruction accesses the cache in order with respect to the compiled sequence of integer (or floating-point) load/store instructions. This means that integer load/stores process in parallel with floating-point load/store instructions.
- An unordered load/store instruction accesses the cache out-of-order with respect to the compiled sequence of any load/store instructions.

Memory disambiguation at compile time marks each load/store instruction.

Branch architecture. Branch-target buffers usually appear in the context of dynamic branch prediction.[25] We try to combine these buffers with static branch prediction. The DSNS architecture assumes the buffers hold the branch-target addresses without prediction bits. It also defines two types of branch instructions with respect to branch prediction:

- A to_enter branch instruction stores its branch-target address in the buffer. A branch instruction can be marked to_enter if and only if it is predicted as taken and its target address is never changed.
- A not_to_enter branch instruction never stores its branch-target address in the buffer. A branch instruction should be marked not_to_enter if it is predicted as not_taken or if its target address can be changed.

Static branch prediction at compile time marks each branch instruction. While a branch-target address is placed in the buffer, the corresponding branch is treated as being predicted as taken at runtime.

Conditional control-flow breaks essentially involve four steps: generating a condition to test, testing the condition to make a branch decision (taken or not taken), computing the branch-target address, and replacing the program counter with the branch-target address if the branch is taken.

Two schemes permit implementation of conditional control transfers in the usual way: compare-and-branch and branch-on-condition-code. The compare-and-branch scheme performs all the previously mentioned steps in one conditional branch instruction and requires no condition codes. The branch-on-condition-code scheme assumes condition codes and separates the first step from the others. Setting a condition code as a side effect of a normal computation accomplishes the first step, and one conditional branch instruction performs all the other steps. We use another scheme, called advanced conditioning,[26] which separates the first and second steps from the third and fourth steps. The first step may be further separated from the second step.

The DSNS architecture introduces new true/false registers and provides additional tags to both general-purpose and floating-point registers. DSNS provides 32 × 1-bit true/false registers. A 4-bit tag for each general-purpose or floating-point register works as a condition code. When an instruction completes execution, it stores the execution state in the tag of its destination register. The DSNS architecture defines the following control transfer instructions (see Figure 8).

- **Test.** This instruction tests the tag of its source register against the specified branch condition and places a Boolean value (true or false) into its destination true/false register. This instruction corresponds to the second step listed earlier.
- **Compare_and_test.** An instruction of this type tests for various relationships between two values in its source registers and places a Boolean value into its destination true/false register. It performs the first and second steps.
- **Branch.** A Boolean value in a source true/false register determines the outcome of a branch (taken or not taken). The hardware generates the branch-target address according to an addressing mode: either relative to the

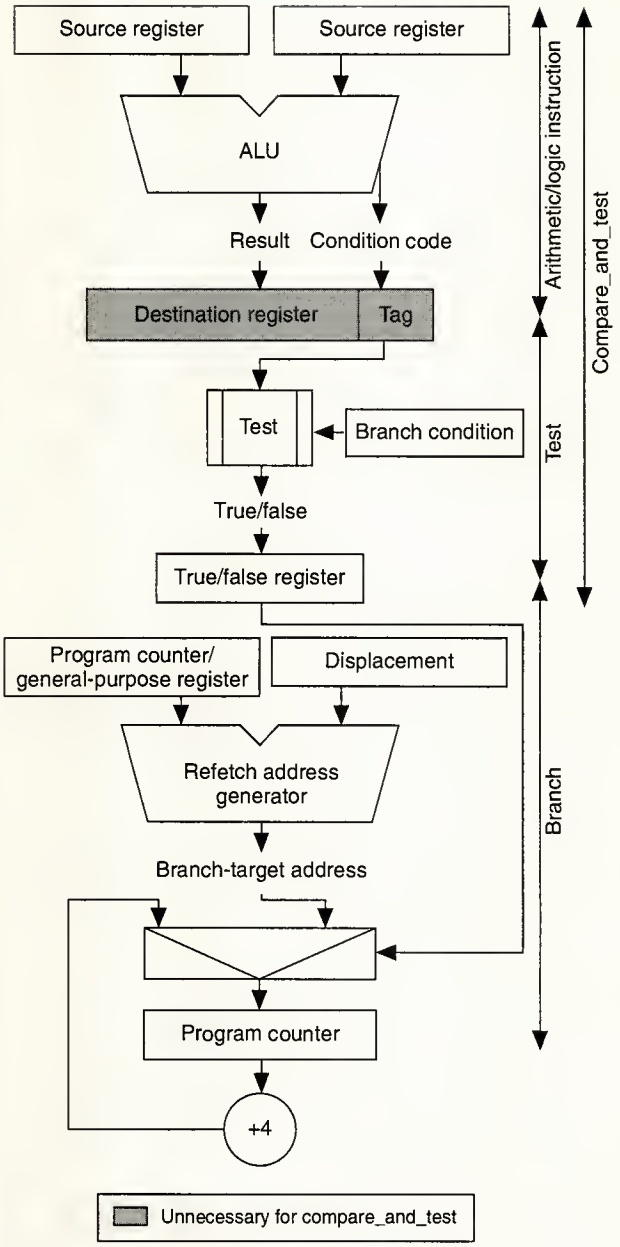


Figure 8. Advanced conditioning.

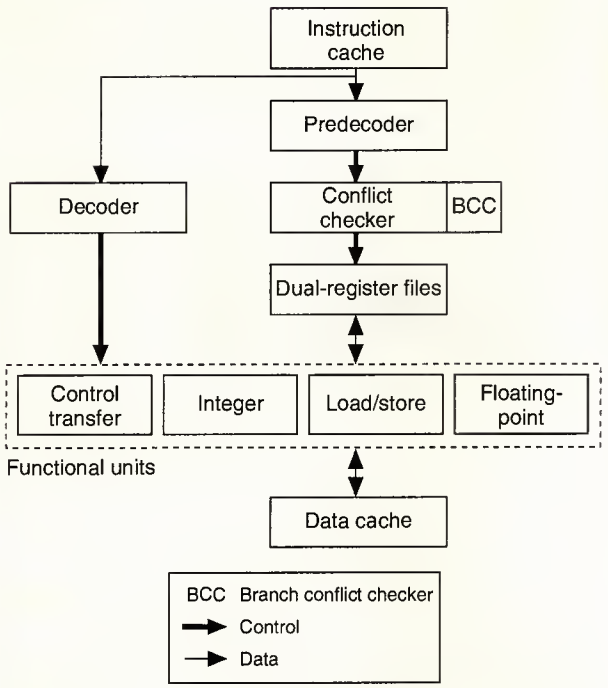


Figure 9. Block diagram of the DSNS processor prototype.

program counter or to the general-purpose register. When a branch is taken, the target address immediately replaces the program counter; that is, it is not a delayed branch. This instruction corresponds to the third and fourth steps.

Furthermore, to deal with multiple conditions, logical instructions operate on two true/false registers.

The advanced conditioning scheme has several advantages. 1) It allows compilers to move test or compare_and_test instructions far up from the branch instruction. 2) It potentially decreases the number of branch instructions to execute because it allows compilers to pack multiple branches into a single branch. 3) It enables hardware to resolve branches early because the branch instruction performs only the address computation. 4) Consequently, this scheme reduces the branch delay and penalty. The AMD Am29000 follows a similar approach.

Dual-register file. Speculative execution such as conditional mode and boosting[22] requires hardware to maintain the precise machine state so that the pipeline can recover from incorrectly predicted branch paths. To facilitate this process, the DSNS processor prototype duplicates its register files (general-purpose, floating-point, and true/false). The functionality of a dual-register file is equivalent to that of the sequential and shadow register files described in Smith et al.;[22] the implementations, however, differ from each other. This means that one could replace our dual-register file with a pair of sequential and shadow register files, and vice versa.

We refer to one file in a dual-register file as RF 0 and to the other as RF 1. A logical register has dual physical locations in both RF 0 and RF 1. Each physical register has three states: current, alternate_valid, and alternate_invalid. Two physical registers for one logical register are mutually exclusive with respect to their currentness; one physical register is exclusively in current state, and the other is exclusively in alternate state.

The dual-register file works as follows:

- **Register read.** An instruction to issue in unconditional mode fetches its source operands from the current registers. An instruction to issue in conditional mode fetches each operand either from the alternate register if it is valid or from the current register otherwise.
- **Register write.** An instruction that completes its execution in unconditional mode writes its result into the current register. An instruction that completes in conditional mode writes its result into the alternate register and validates it.
- **Branch resolution.** If the branch that has caused the conditional mode is correctly predicted, all the valid alternate registers change their states to current; the counterparts will be in alternate_invalid states. The other registers remain unchanged. An incorrect prediction, on the other hand, invalidates all the alternate registers.

The degree of duplication for register files depends on the degree of conditional modes. Our dual-register file can buffer all side effects produced through one conditional mode.

Hardware overview. As our future work, we plan a VLSI implementation of the DSNS architecture. We are currently implementing the prototype in off-the-shelf transistor-transistor logic and CMOS building blocks. The clock cycle time is 60 ns, and the instruction pipeline divides into four stages: IF (instruction fetch), D (decode), E (execute), and W (write back). Four instructions proceed in parallel.

Figures 9 and 10 illustrate the overall structure and the data path configuration of the DSNS processor prototype. The prototype consists of an instruction cache, predecoder, decoder, conflict checker, branch unit, and functional units.

The instruction cache is a 64-byte-line, 512-Kbyte virtual address, direct-mapped cache. To fetch four instructions (called a fetch block) at a time, the prototype provides a 128-bit-wide cache. The cache consists of approximately 8,000 lines and 32,000 fetch blocks.

To allow the instruction fetch to continue, a branch-target buffer shares the directory (tag memory) with the instruction cache. This buffer consists of approximately 32,000 entries, each of which is associated with a fetch block. Each buffer

entry contains a predicted target address and a word offset within the fetch block. The buffer can be accessed twice in one cycle, first for a branch prediction and then for storing a branch-target address. The fetched instructions are immediately predecoded to obtain the information necessary to check for hazards.

In the decoder, instructions are completely decoded using ROM to obtain the full information necessary to control the functional units.

The conflict checker checks for structural and read-and-write and write-and-write data hazards, according to the outcome of the predecoder.

The branch unit is a three-stage pipeline responsible for the execution of branch instructions. It also provides the conflict checker with the branch instruction that steps aside and waits for the hazard resolution.

As shown in Figure 10, the DSNS prototype contains four operational units each of which contain a variety of independent functional units. The integer unit contains two ALUs, a shifter, and a multiplier; the floating-point unit contains an ALU, a multiplier, a divider, and a converter; the load/store unit includes two identical units; and the control transfer unit contains a refetch-address generator and two units for advanced conditioning.

Except for the floating-point divider and the functional units with result latency of one cycle, all the functional units are pipelined. Although not shown in Figure 10, the data path contains bypasses from the output buses to the register-file read ports for operand forwarding.

As shown in Figure 10, the data path includes three user-visible register files, each of which is a dual-register file:

- True/false 32 × 1-bit registers provide three read ports and one write port.
- Each 32 × 32-bit general-purpose

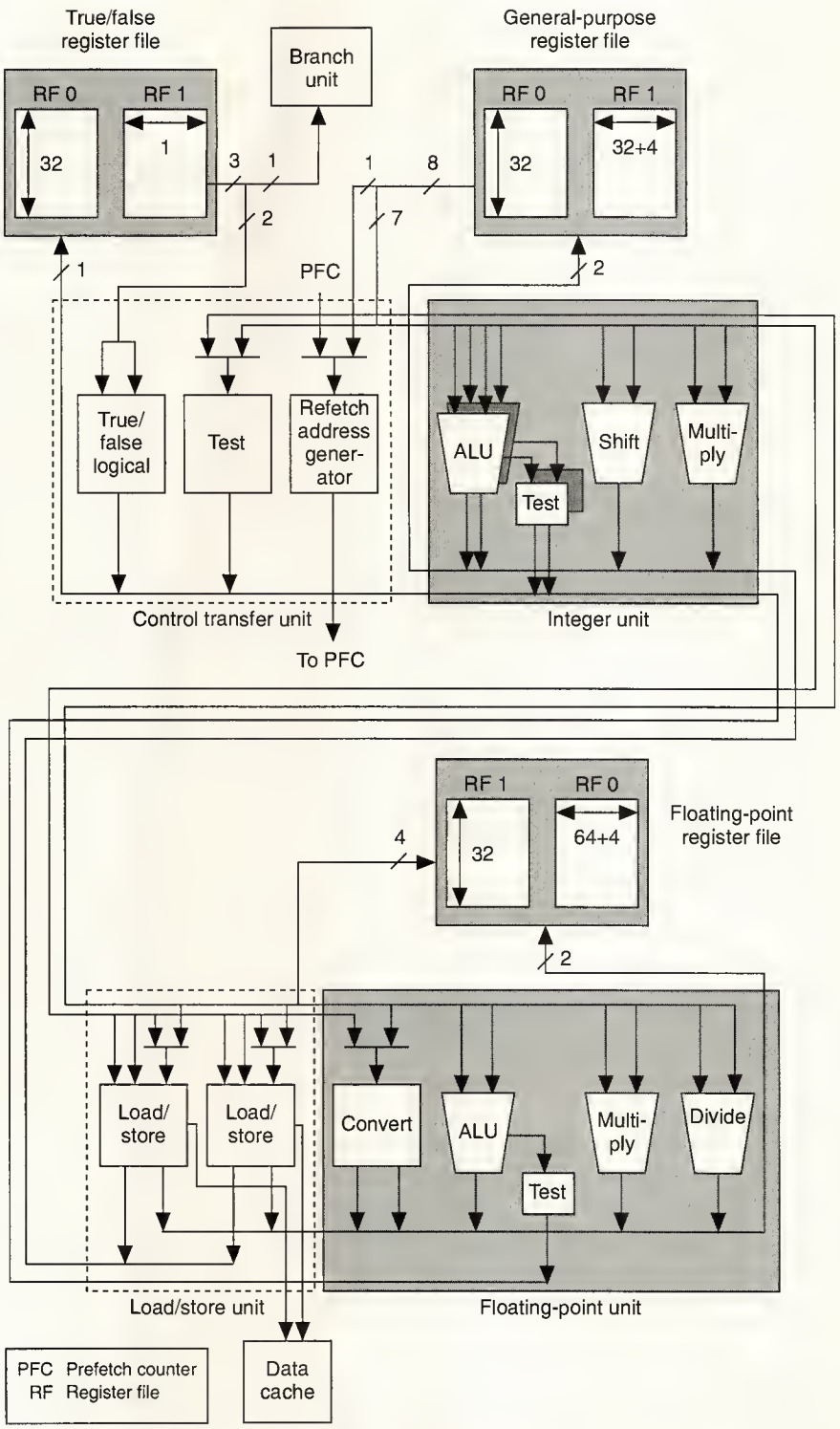


Figure 10. Block diagram of the data path in the DSNS processor prototype.

register with its 4-bit tag provides eight read ports and two write ports.
- Each 32 × 64-bit floating-point register file with its 4-bit tag provides four read ports and two write ports.

DSNS processors access dual-register files twice in one cycle, first for register writes and then for register reads.

We configured the dual-port, 64-byte-line, 512-Kbyte data cache as a virtual-address, direct-mapped, nonblocking cache. It provides two 64-bit-wide ports with two load/store units. The directory (tag array) is duplicated twice for two ports, but two ports share the data array, which consists of two individual 64-bit-wide banks. The pipelined cache access advances in two stages, tag lookup and data-array access.

THE KRPP AND THE DSNS SUPERSCALAR architecture exploit parallelism at task and instruction levels respectively. We developed each system separately; that is, the KRPP under development comprises multiple microprocessors rather than the DSNS processors. In the future, we plan to replace the microprocessors in the KRPP with superscalar processors. Currently, both projects are in the final stage of logic design, and we expect to complete them this December.

Parallel computers consisting of some hundreds of superscalar (or VLIW) processors, each of which can perform instruction-level parallelism, suit parallel processing that comprises different function levels and computational models. We believe they will become the key architecture in the near future.

We have several subprojects in progress: the construction of the parallel programming environment—including a parallel programming language, parallel program debuggers, and a parallelizing compiler we call the IPC, or Integrated Parallelizing Compiler—and a network synthesis system for the reconfigurable parallel processor system.

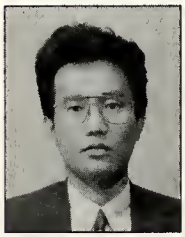
Acknowledgments

We thank the students who have contributed and those who are now contributing to the projects. We also acknowledge the considerable contributions from N. Sakuta with Asia Electronics Inc., M. Takamura and K. Uchida of Fujitsu Ltd., and S. Oyanagi and N. Tanabe of Toshiba Corp.

References

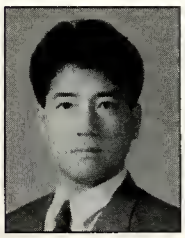
1. K. Murakami et al., "An Overview of the Kyushu University Reconfigurable Parallel Processor," *ACM Sigarch Computer Architecture News,* Vol. 16, No. 4, Sept. 1988, pp. 130-137.
2. K. Murakami et al., "The Kyushu University Reconfigurable Parallel Processor: Design of Memory and Intercommunication Architectures," *Proc. ACM Sigarch Int'l Conf. Supercomputing,* June 1989, pp. 351-360.
3. K. Murakami et al., "The Kyushu University Reconfigurable Parallel Processor: Design Philosophy and Architecture," *Proc. IFIP 11th World Computer Congress,* Aug. 1989, pp. 995-1000.
4. S. Mori et al., "The Kyushu University Reconfigurable Parallel Processor: Cache Architecture and Cache Coherence Schemes," *Proc. Int'l Symp. Shared-Memory Multiprocessing,* Apr. 1991, pp. 218-229.
5. K. Murakami et al., "SIMP (Single Instruction Stream/Multiple Instruction Pipelining): A Novel High-Speed Single-Processor Architecture," *Proc. 16th Int'l Symp. Computer Architecture,* May 1989, pp. 78-85.
6. M.J. Flynn, "Very High-Speed Computing Systems," *Proc. IEEE,* Vol. 54, No. 12, Dec. 1966, pp. 1901-1909.
7. H.J. Siegel, R.J. McMillen, and P.T. Mueller, "A Survey of Interconnection Methods for Reconfigurable Parallel Processing System," *Proc. Nat'l Computer Conf.,* Vol. 48, June 1979, pp. 529-542.
8. C.R. Vick, S.P. Kartashev, and S.I. Kartashev, "Adaptable Architectures for Supersystems," *Computer,* Vol. 13, No. 11, Nove. 1980, pp. 17-35.
9. S. Yalamanchili and J.K. Aggarwal, "Reconfiguration Strategies for Parallel Architecture," *Computer,* Vol. 18, No. 12, Dec. 1985, pp. 44-61.
10. R.J. Swan, S.H. Fuller, and D.P. Siewiorek, "Cm*—A Modular, Multi-microprocessor," *Proc. Nat'l Computer Conf.,* Vol. 46, 1977, pp. 637-644.
11. W.C. Brantley, K.P. McAuliffe, and J. Weiss, "RP3 Processor-Memory Element," *Proc. Int'l Conf. Parallel Processing,* Aug. 1985, pp. 782-789.
12. R. Rettberg and R. Thomas, "Contention Is No Obstacle to Shared-Memory Multiprocessing," *Comm. ACM,* Vol. 29, No. 12, Dec. 1986, pp. 1202-1212.
13. M. Accetta et al., "Mach: A New Kernel Foundation for Unix Development," *Proc. Usenix Summer Conf.,* 1986, pp. 93-112.
14. A. Tucker and A. Gupta, "Process Control and Scheduling Issues for Multiprogrammed Shared-Memory Multiprocessors," *Proc. 12th ACM Symp. Operating Systems Principles,* 1989, pp. 159-166.
15. D.L. Black, "Scheduling Support for Concurrency and Parallelism in the Mach Operating System," *Computer,* Vol. 23, No. 5, 1990, pp. 35-43.
16. N.P. Jouppi, "Available Instruction-Level Parallelism for Superscalar and Superpipelined Machines," *Proc. Third Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS-III),* Apr. 1989, pp. 272-282.
17. J.E. Smith, "Dynamic Instruction Scheduling and the Astronautics ZS-1," *Computer,* Vol. 22, No. 7, July 1989, pp. 21-35.
18. S. Weiss and J.E. Smith, "Instruction Issue Logic in Pipelined Supercomputers," *IEEE Trans. Computers,* Vol. C-33, No. 11, Nov. 1984, pp. 1013-1022.

19. J.E. Thornton, "Parallel Operation in the Control Data 6600," *Proc. Fall Joint Computer Conf.*, Vol. 26, 1964, pp. 33-40.
20. R.M. Tomasulo, "An Efficient Algorithm for Exploiting Multiple Arithmetic Units," *IBM J. Res. Dev.*, Vol. 11, Jan. 1967, pp. 25-33.
21. N.P. Jouppi, "The Nonuniform Distribution of Instruction-Level and Machine Parallelism and Its Effect on Performance," *IEEE Trans. Computers*, Vol. 37, No. 12, Dec. 1989, pp. 1645-1658.
22. M.D. Smith, M.S. Lam, and M.A.Horowitz, "Boosting Beyond Static Scheduling in a Superscalar Processor," *Proc. 17th Int'l Symp. Computer Architecture*, May 1990, pp. 344-354.
23. S. Tomita et al., "A Dynamically Microprogrammable Computer with Low-Level Parallelism," *IEEE Trans. Computers*, Vol. C-29, No. 7, July 1980, pp. 577-595.
24. S. Tomita et al., "A Computer with Low-Level Parallelism: Its Application to 3D Graphics and Prolog/Lisp Machines," *Proc. 13th Int'l Symp. Computer Architecture*, 1986, pp. 280-289.
25. J.K.F. Lee and A.J. Smith, "Branch Prediction Strategies and Branch Target Buffer Design," *Computer*, Vol. 17, No. 1, Jan. 1984, pp. 6-22.
26. W.G. Rosocha and E.S. Lee, "Performance Enhancement of SISD Processors," *Proc. Sixth Int'l Symp. Computer Architecture*, Apr. 1979, pp. 216-231.



Akira Fukuda is an associate professor in the Department of Information Systems at Kyushu University. Since 1986 he has directed the parallel operating system and parallel programming language projects in the KRPP project. Previously he worked for the Nippon Telegraph and Telephone Corporation, where he engaged in research on performance evaluation of computer systems and queueing theory. His research interests include parallel and distributed operating systems, parallel processing, and performance evaluation of computer systems.

Fukuda holds BE, ME, and DrEng degrees in computer science and communication engineering from Kyushu University. He is a member of the IEEE Computer Society, Association of Computing Machinery, IEICE (Institute of Electronics, Information, and Communication Engineers) of Japan, IPSJ (Information Processing Society of Japan), and Operations Research Society of Japan.



Kazuaki Murakami is also an assistant professor in the Department of Information Systems at Kyushu University. Since 1987 he has directed the hardware and compiler projects described in this article. Previously, he worked for Fujitsu Limited as a computer architect of the FACOM M-series mainframe computers. His current research interests include computer architecture, parallel processing, and the parallelizing compiler.

Murakami received the BE and ME degrees in computer science and engineering from Kyoto University. He has served as a director of the IPSJ Special Interest Group on Computer Architecture for the past two years and as a secretary of SWOPP (Summer Workshop on Parallel Processing) for the past four years. He is a member of the IEEE, IEEE Computer Society, ACM, IEICE, IPSJ, and JSIAM (Japan Society for Industrial and Applied Mathematics).

Shinji Tomita is a professor at Kyoto University. From 1986 to 1991 while at Kyushu University, he engaged in designing the reconfigurable parallel processor and the superscalar processor introduced in this article. Earlier at Kyoto University, he had developed the QA-1 and QA-2 machines on which he pioneered work on VLIW machine architecture. His current interest is in the architecture of an ultrascale multiprocessor system.

Tomita received the BE, ME, and DrEng degrees from Kyoto University. He currently serves as chair of the IPSJ Special Interest Group on Computer Architecture and as chair of two annual conferences on parallel processing held in Japan, the JSPP (Joint Symposium on Parallel Processing) and SWOPP in which more than 200 researchers participate. He is a member of the IEICE, IPSJ, ACM, and IEEE Computer Society.

Address questions concerning the parallel operating system in this article to Akira Fukuda and the architectures to Kazuaki Murakami. Contact both authors at the Department of Information Systems, Interdisciplinary Graduate School of Engineering Sciences, Kyushu University, 6-1 Kasuga-koen, Kasuga-shi, Fukuoka, 816 Japan; or e-mail at fukuda@is.kyushu-u.ac.jp and murakami@is.kyushu-u.ac.jp.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 153 Medium 154 High 155

Gmicro/100

continued from p. 23

does not exchange them in the contrary case. If an incorrect prediction takes the prebranch, the conditional branch instruction takes another branch to its next instruction in the E stage.

Figure 4 details the pipeline time sequences of the conditional branch instruction (branch not equal, or BNE) and its branch target instruction. An unsuccessful conditional branch instruction with a correct prediction takes two clock cycles (4a). A successful conditional branch instruction with an incorrect prediction (4b) and an unsuccessful conditional branch instruction with an incorrect prediction (4c) execute in nine clock cycles. Executing a successful condition branch instruction with a correct prediction (4d) takes five clock cycles.

Without the prebranch, an unsuccessful conditional branch instruction takes two clock cycles; its pipeline time sequence is the same as found in Figure 4a. A successful conditional branch instruction without the prebranch executes in nine clock cycles; its pipeline time sequence is similar to Figure 4b.

The BNE instruction executes in five clock cycles when the successful branch BNE instruction takes a prebranch in the D stage. It executes in nine clock cycles if the successful branch BNE instruction does not take the prebranch. When the prediction is correct, the prejump mechanism restores four clock cycles for the successful branch BNE instruction. The unsuccessful branch BNE instruction that takes an incorrect prebranch executes in nine clock cycles, or two clock cycles if it does not take the prebranch. When the prediction is not correct, the prejump mechanism wastes seven clock cycles for the unsuccessful branch BNE instruction. The prebranch taken by the unsuccessful conditional branch instruction reduces the processor performance.

The prebranch for the conditional branch instruction does not always restore the lost performance. Table 2 lists parameters for conditional branch instructions.

We can calculate the average numbers of clock cycles, *Bnp* and *Bp*, by

$$Bnp = b\text{Neb} + (1 - b)\text{Nnb}$$
$$Bp = b[p\text{Npb} + (1 - p)\text{Neb}] + (1 - b)(q\text{Nnb} + (1 - q)\text{Neb})$$

If *Bp* is smaller than *Bnp*, the prejump mechanism is effective for the conditional branch instruction. This condition is obtained by the inequality

$$(\text{Neb} - \text{Npb})bp + (\text{Neb} - \text{Nnb})(1 - b)(q - 1) > 0$$

For the Gmicro/100, Nnb equals 2, Npb equals 5, and Neb equals 9. When the probability that the conditional branch

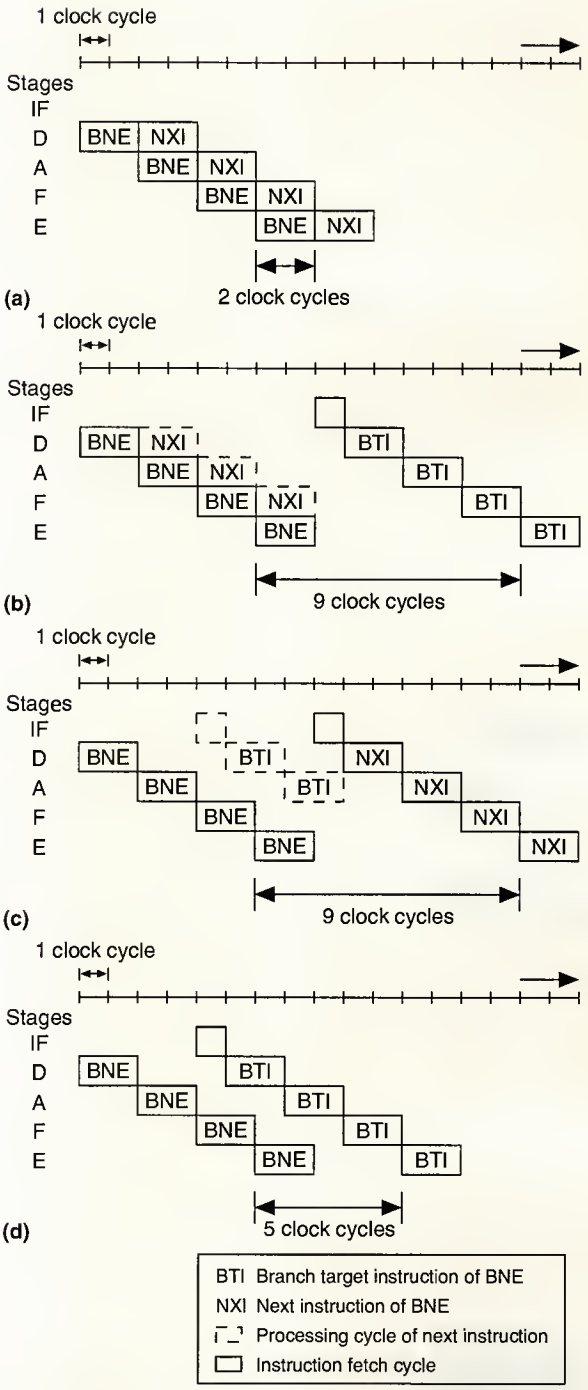


Figure 4. Pipeline time sequences for BNE execution: unsuccessful branch and correct prediction (a), successful branch and incorrect prediction (b), unsuccessful branch and incorrect prediction (c), and successful branch and correct prediction (d).

instruction takes a branch is 50 percent, b equals 0.5, the inequality becomes $4 / 7 \times p + q > 1$.

To simplify the discussion, let us suppose p equals q. Then, the inequality becomes $p > 7 / 11$.

The prejump is effective for restoring the lost clock cycles when the prediction correction rate is more than 64 percent—a reasonable number. But p and q are not equal; sometimes they are quite different. Therefore, the taking of a prebranch by a conditional branch instruction does not always improve processor performance.

Return-from-subroutine instruction. The jump target addresses of PC-relative jump instructions are static (fixed at compiling time), but the jump target addresses of return-from-subroutine instructions are dynamic. The jump target address of the return-from-subroutine instruction resides in the stack area of the memory. Also, the stack top is not fixed before the execution of all the instructions proceeding the return-from-subroutine instruction. The prejump mechanism for the return-from-subroutine instruction differs from the PC-relative jump instructions (the unconditional and conditional branch instructions).

The Gmicro/100 implements new hardware called the PC stack. The return-from-subroutine instruction takes the prejump (the prereturn) by using this special hardware (see Figure 5). The PC stack, an eight-word stack memory on a chip, holds copies of the program counter values pushed to the stack area in the off-chip memory upon execution of jump-to-subroutine instructions. When decoding of a return-from-subroutine instruction occurs in the D stage, the address at the top of the PC stack "is popped" (fetched) and the instruction flow sequence changes to the popped address. The address at the stack top in the off-chip memory is fetched when a return-from-subroutine instruction is processed in the F stage. The address fetched from the off-chip memory is the actual return address.

Table 2. Conditional branch parameters

Parameter	Description
b	Probability that a conditional branch instruction takes a branch
p	Probability that a prediction is correct (that the branch will be taken)
q	Probability that a prediction is correct (that the branch will not be taken)
Nnb	Number of clock cycles required for an unsuccessful conditional branch instruction that does not take a prebranch in the D stage (two clock cycles for the Gmicro/100)
Npb	Number of clock cycles required for a successful conditional branch instruction that takes a prebranch in the D stage (five clock cycles for the Gmicro/100)
Neb	Number of clock cycles required for an unsuccessful conditional branch instruction that takes a prebranch in the D stage or a successful conditional branch instruction that does not take a prebranch in the D stage. (In this case, a branch is taken in the E stage and it needs nine clock cycles for the Gmicro/100.)
Bnp	Average number of clock cycles required for a conditional branch instruction in the event that no prebranch is taken
Bp	Average number of clock cycles required for a conditional branch instruction in the event that the prebranch is taken depending on the branch prediction with the probability *p* and *q*

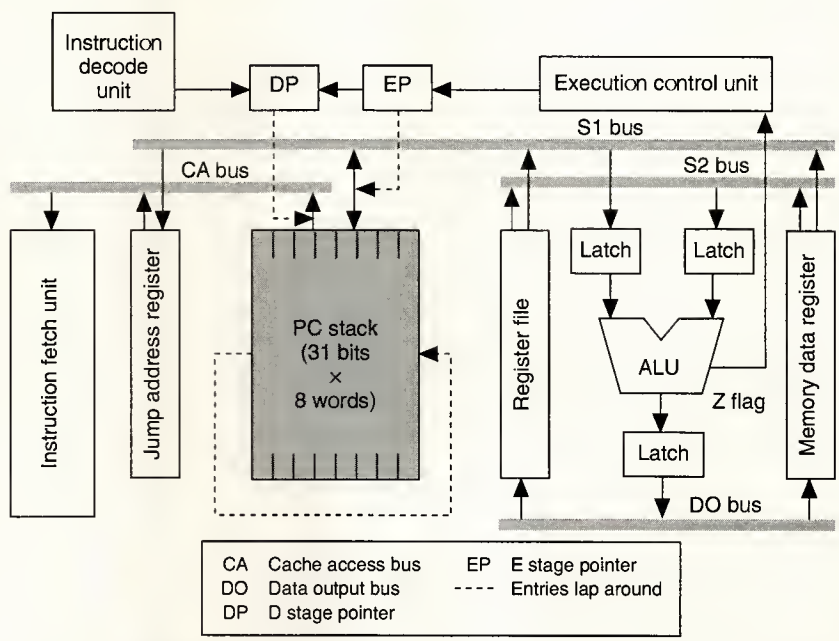


Figure 5. Diagram of the prereturn mechanism.

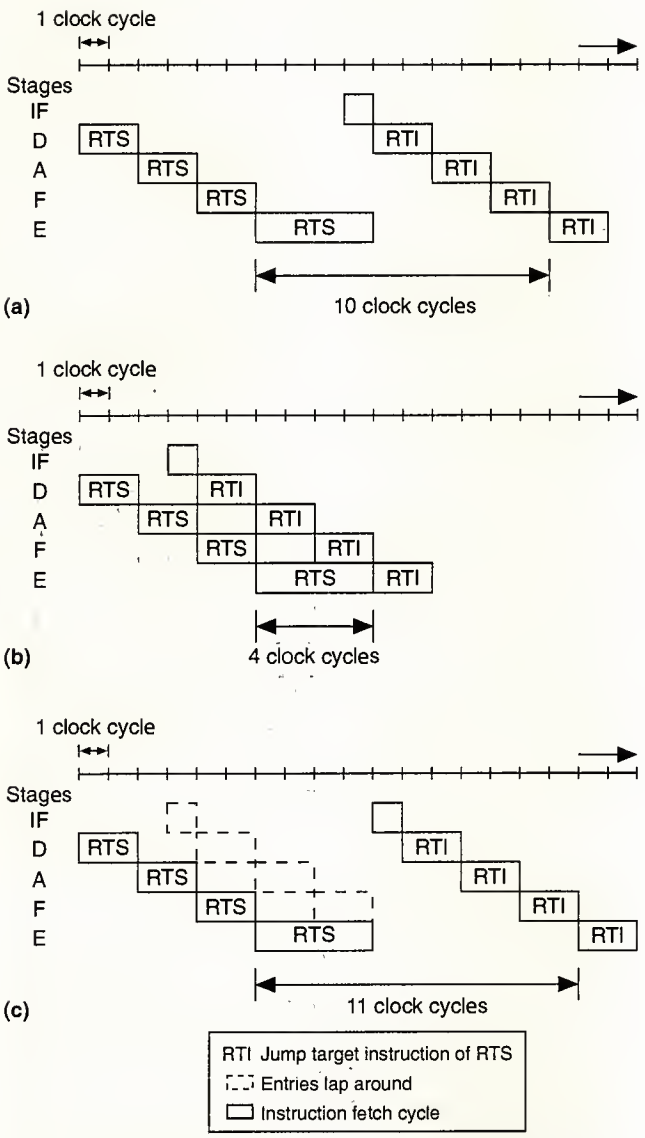


Figure 6. Pipeline time sequences for RTS execution: without prereturn (a), correct prereturn target address (b), and incorrect prereturn target address (c).

The address used for the prereturn's target address compares with the address popped from the off-chip memory by the ALU in the E stage. If the two addresses are of the same value, no jump is taken by the return-from-subroutine instruction in the E stage. If the two addresses differ, a jump to the actual return address popped from the off-chip memory is taken in the E stage (it means that the prereturn is not correct).

The prereturn is not correct, for example, when a context switch occurs during the execution of the subroutine (then the stack pointer will change). If the nest level of the subroutines is more than the eighth level, older entries in the PC stack are overwritten. In such a case, the prereturns are incorrect, except for the current eight levels. But in almost all cases the prereturn is correct, and the PC stack effectively reduces the performance degradation caused by the return-from-subroutine instruction.

Figure 6 details the pipeline time sequences of the return-from-subroutine (RTS) instruction and its jump target instruction. When the return-from-subroutine instruction takes a correct prereturn in the D stage, the instruction executes in four clock cycles. When the prereturn is not correct (returns to the wrong address), it takes 11 clock cycles to take a correct return in the E stage. When there is no prereturn and the return-from-subroutine instruction always takes a return in the E stage, it executes in 10 clock cycles. The prejump mechanism restores six clock

Table 3. Return-to-subroutine instruction parameters.

Parameter	Description
a	Probability that a target address used for the prereturn is correct
Nnpr	Number of clock cycles required for a return-from-subroutine instruction without the prereturn (10 clock cycles for the Gmicro/100 in the case of the RTS instruction)
Npr	Number of clock cycles required for a return-from-subroutine instruction that takes a prereturn in the D stage (four clock cycles for the Gmicro/100 in the case of the RTS instruction)
Ner	Number of clock cycles required for a return-from-subroutine instruction that takes a incorrect prereturn in the D stage and takes a return in the E stage again (11 clock cycles for the Gmicro/100 in the case of the RTS instruction)
Rnp	Average number of clock cycles required for a return-from-subroutine instruction when no prereturn is taken
Rp	Average number of clock cycles required for a return-from-subroutine instruction in the case that the prereturn is taken with the probability *a*

SUBSCRIPTION CARD READER-SERVICE CARD READER-SERVICE CARD

IEEE MICRO

August 1991 issue (card void after February 1992)

Name _____

Title _____

Company _____

Address _____

City _____ State _____ Zip _____

Country _____ Phone(_____) _____

Reader Interest

(Add comments on the back)

Readers,
Indicate your interest in articles and departments **by circling the appropriate number** (shown on the last page of articles and departments):

150	151	152	180	181	182
153	154	155	183	184	185
156	157	158	186	187	188
159	160	161	189	190	191
162	163	164	192	193	194
165	166	167	195	196	197
168	169	170	198	199	200
171	172	173	201	202	203
174	175	176	204	205	206
177	178	179	207	208	209

Product Information 1

(Circle the numbers to receive product information)

1	21	41	61	81	101	121	141
2	22	42	62	82	102	122	142
3	23	43	63	83	103	123	143
4	24	44	64	84	104	124	144
5	25	45	65	85	105	125	145
6	26	46	66	86	106	126	146
7	27	47	67	87	107	127	147
8	28	48	68	88	108	128	148
9	29	49	69	89	109	129	149
10	30	50	70	90	110	130	
11	31	51	71	91	111	131	
12	32	52	72	92	112	132	
13	33	53	73	93	113	133	
14	34	54	74	94	114	134	
15	35	55	75	95	115	135	
16	36	56	76	96	116	136	
17	37	57	77	97	117	137	
18	38	58	78	98	118	138	
19	39	59	79	99	119	139	
20	40	60	80	100	120	140	

IEEE MICRO

August 1991 issue (card void after February 1992)

Name _____

Title _____

Company _____

Address _____

City _____ State _____ Zip _____

Country _____ Phone(_____) _____

Reader Interest

(Add comments on the back)

Readers,
Indicate your interest in articles and departments **by circling the appropriate number** (shown on the last page of articles and departments):

150	151	152	180	181	182
153	154	155	183	184	185
156	157	158	186	187	188
159	160	161	189	190	191
162	163	164	192	193	194
165	166	167	195	196	197
168	169	170	198	199	200
171	172	173	201	202	203
174	175	176	204	205	206
177	178	179	207	208	209

Product Information 2

(Circle the numbers to receive product information)

1	21	41	61	81	101	121	141
2	22	42	62	82	102	122	142
3	23	43	63	83	103	123	143
4	24	44	64	84	104	124	144
5	25	45	65	85	105	125	145
6	26	46	66	86	106	126	146
7	27	47	67	87	107	127	147
8	28	48	68	88	108	128	148
9	29	49	69	89	109	129	149
10	30	50	70	90	110	130	
11	31	51	71	91	111	131	
12	32	52	72	92	112	132	
13	33	53	73	93	113	133	
14	34	54	74	94	114	134	
15	35	55	75	95	115	135	
16	36	56	76	96	116	136	
17	37	57	77	97	117	137	
18	38	58	78	98	118	138	
19	39	59	79	99	119	139	
20	40	60	80	100	120	140	

SUBSCRIBE TO IEEE MICRO

All the facts about today's chips and systems

❑ **YES**, sign me up!

If you are a member of the Computer Society or any other IEEE society, pay the member rate of only $21 for a year's subscription (six issues).

Society: _____

IEEE membership no: _____

Society members: Subscriptions are annualized. For orders submitted March through August, pay half the full-year rate ($10.50) for three bimonthly issues.

_____
Full Signature Date

_____
Name

_____
Street

_____
City

_____
State/Country ZIP/Postal Code

❑ **YES**, sign me up!

If you are a member of ACM, ACS, BCS, IEE (UK), IEEE but not a member of an IEEE society), IECEJ, IPSJ, NSPE, SCS, or other professional society, pay the sister-society rate of only $38 for a year's subscription (six issues).

Organization: _____

Membership no: _____

❑ Payment enclosed *DC residents: add sales tax*

❑ Charge to ❑ Visa ❑ MasterCard ❑ American Express

Charge-card number

_____

Expiration date

____ ____
Month *Year*

Prices valid through 12/31/91

8/91 MICRO

Charge orders also taken by phone:
(714) 821-8380 8 a.m. to 5 p.m. Pacific time
Circulation Dept.
10662 Los Vaqueros Circ., PO Box 3014
Los Alamitos, CA 90720-1264

Editorial comments

I liked: _____

_____

_____

I disliked: _____

_____

_____

I would like to see: _____

_____

_____

Reviewers Needed. *If interested, send professional data to Dante Del Corso, Dipartimento di Elettronica, Politecnico di Torino, C.so Duca degli Abruzzi, 24, 10129, Torino, Italy.*

For reader-service inquiries, see other side.

PLACE POSTAGE HERE

PO Box is for reader-service cards only.

IEEE Micro
PO BOX 16508
NORTH HOLLYWOOD CA 91615-6508
USA

Editorial comments

I liked: _____

_____

_____

I disliked: _____

_____

_____

I would like to see: _____

_____

_____

Reviewers Needed. *If interested, send professional data to Dante Del Corso, Dipartimento di Elettronica, Politecnico di Torino, C.so Duca degli Abruzzi, 24, 10129, Torino, Italy.*

For reader-service inquiries, see other side.

PLACE POSTAGE HERE

PO Box is for reader-service cards only.

IEEE Micro
PO BOX 16508
NORTH HOLLYWOOD CA 91615-6508
USA

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 38 LOS ALAMITOS, CA

POSTAGE WILL BE PAID BY ADDRESSEE

IEEE COMPUTER SOCIETY
PO BOX 3014
LOS ALAMITOS CA 90720-9804
USA

Table 4. Bitmap instructions.

Instruction	Function	Parameters in registers
BVMAP	A Boolean operation specified by the lower 4 bits of the register R5 is performed between the source bit string and the destination bit string. The resultant bit string is stored in the destination bit string field.	R0 Base address of the source bit string R1 Offset of the source bit string R2 Length of the bit strings R3 Base address of the destination bit string R4 Offset of the destination bit string R5 Type of Boolean operation (lower 4 bits)
BVCPY	The source bit string is copied to the destination bit string field.	R0 Base address of the source bit string R1 Offset of the source bit string R2 Length of the bit strings R3 Base address of the destination bit string R4 Offset of the destination bit string
BVPAT	A Boolean operation specified by the lower 4 bits of the register R5 is performed between the source bit string in register R0 and the destination bit string. The resultant bit string is stored into the destination bit string field.	R0 Source bit string R1 Don't care R2 Length of the destination bit string R3 Base address of the destination bit string R4 Offset of the destination bit string R5 Type of Boolean operation (lower 4 bits)

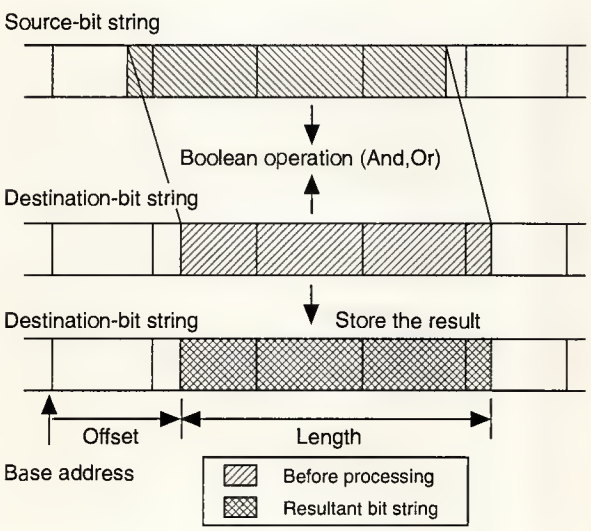


Figure 7. Function of bitmap instructions.

cycles for the return-from-subroutine instruction when the prereturn is correct. When the prereturn is not correct, one clock cycle is lost.

Table 3 provides parameters for return-to-subroutine instructions. We can calculate the average numbers of clock cycles, *Rnp* and *Rp*, by

$$Rnp = \text{Nnpr}$$
$$Rp = a\text{Npr} + (1 - a)\text{Ner}$$

If *Rp* is smaller than *Rnp*, the prejump mechanism is effective for the return-from-subroutine instruction. This condition is obtained by the inequality $a > (\text{Ner} - \text{Nnpr}) / (\text{Ner} - \text{Npr})$

For the Gmicro/100, Nnpr equals 10, Npr equals 4, and Ner equals 11 in the case of the RTS instruction. Then, the inequality becomes $a > 1 / 7$.

This condition can be realized by using a small-capacity PC stack. The return-from-subroutine instruction always takes a jump. Even when the prereturn is not taken by the return-from-subroutine instruction in the D stage, the instruction always takes a jump.

Bitmap manipulation

The Gmicro/100 has three bitmap manipulation instructions to support bit-block transfer primitives. The base address, bit offset, and bit length specify the position of a bit string. General-purpose registers keep the parameters.[11] Table 4 lists the bitmap instructions.

Figure 7 shows an example of bitmap-manipulation function of the BVMAP instruction. The Gmicro/100 reads the

source-bit string and the destination-bit string from memory, performs a Boolean operation on each pair of bits, and stores the resultant bit string into the destination-bit string field.

Interrupt handling during the bitmap processing. A bitmap instruction manipulates very long bit strings with a maximum length of 4 Gbits. We need to accept external interrupts during the processing of the bitmap instruction. The Gmicro/100 checks external interrupts at the time it finishes the processing of every 32-bit block of a bit string. Updating the offset and length of the bit string occurs every time the Gmicro/100 checks external interrupts.

If the Gmicro/100 accepts an external interrupt during the processing of the bitmap instruction, it suspends the processing, saves the processor status word and the address of the bitmap instruction into the stack, and invokes the interrupt handler. The general-purpose registers hold the offset and the length of the bit string that has not yet been processed.

The microprocessor executes the return-from-interrupt instruction at the end of the interrupt handler. This instruction restores the processor status word and returns to the suspended bitmap instruction. The bitmap instruction continues the processing by using the offset and the length of the rest of the bit string. The interrupt handler saves and restores the bitmap instruction parameters, which are kept in general-purpose registers.

Table 5. Basic instruction execution times.

Type	Mnemonic	Operation		Clock count
Short format	MOV R,R	Move	Reg to Reg	2
	MOV M,R	Move	Mem to Reg	2
	MOV R,M	Move	Reg to Mem	2
	CMP M,R	Compare	Mem with Reg	2
	ADD M,R	Add	Mem to Reg	2
	SHL #exp,M	Shift	logical left or right	2
Long format	MOV M,M	Move	Mem to Mem	4
	CMP M,M	Compare	Mem with Mem	4
	ADD M,M	Add	Mem to Mem	4
Jump	BRA label	Branch always		5*
	BCC label	Branch conditionally (taken)		5*
	BCC label	Branch conditionally (not taken)		2*
	BSR label	Branch to subroutine		5*
	RTS label	Return from subroutine		4*

* In case the prejump is correct, and the cache hits. If the cache misses, it takes an additional cycle.

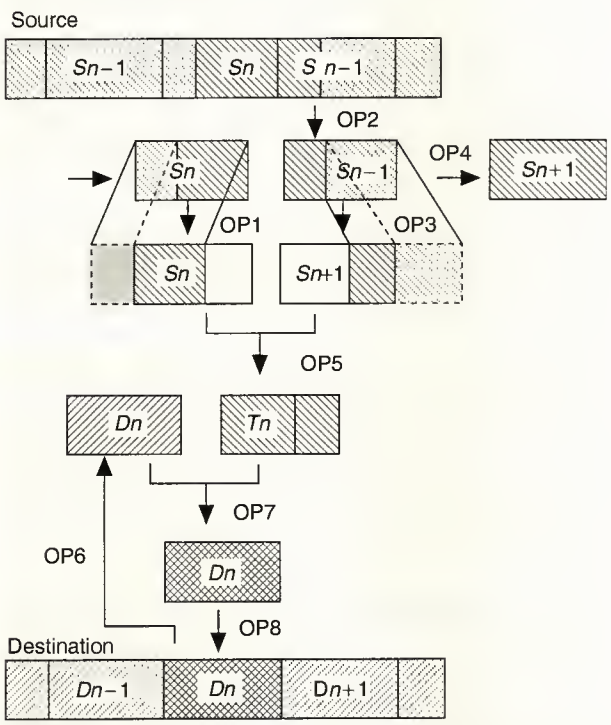


Figure 8. Microoperation loop in the BVMAP execution sequence.

Pipelining of the microoperations. The Gmicro/100 achieves fast execution of the bitmap instructions by pipelining the microoperations in the E stage.[3] Figure 8 shows the microoperation loop in the BVMAP instruction execution sequence. The 32-bit block repeatedly processes the source-bit and destination-bit strings. The loop consists of eight microoperations: OP1 and OP3 are shift operations; OP2, OP6, and OP8 are memory access operations; and OP4, OP5, and OP7 are ALU operations.

A directly programmed loop takes five microinstruction steps. We program the nth cycle of the loop as follows:

Step 1 OP1(n) and OP2(n)
Step 2 OP3(n) and OP4(n)
Step 3 OP5(n) and OP6(n)
Step 4 OP7(n)
Step 5 OP8(n)

We unroll the loop, pipeline the microoperations of the nth and the $(n + 1)$th cycles of the loop, and increase the number of parallel operations executed by one microinstruction. Then, it takes three microinstruction steps to program the loop. We construct the optimized loop as follows:

Step 1 OP7($n - 1$), OP1(n), and OP2(n)

Step 2	OP8($n-1$), OP3(n), and OP4(n)
Step 3	OP5(n) and OP6(n)
Step 4	OP7(n), OP1($n+1$), and OP2($n+1$) = Step 1
Step 5	OP8(n), OP3($n+1$), and OP4($n+1$) = Step 2

The three-step loop is optimum because the memory bus is busy during every step. The Gmicro/100 executes the BVMAP instruction at the microprocessor's maximum bus speed.

We optimize the BVCPY and BVPAT instructions by the same method and achieve optimum use of the memory bus.

Performance evaluation

Table 5 details the clock cycles of the basic instructions, which are achieved when the pipeline is fully operational. The pipeline scheme makes it possible for the execution clock cycles to avoid relying on the addressing mode of the operand. The peak performance of simple instructions is 16.7 MIPS at a 33.3-MHz clock rate.

The Gmicro/100 executes 26.7K Dhrystones per second at 33.3 MHz. We compiled the Dhrystone Version 2.1 program by the GHS C compiler Version 1.8.5 to measure the performance.[12] We used the in-lining optimization option of the GHS C compiler when we compiled the Dhrystone program.[13]

Table 6 shows the number of execution steps of three bitmap instructions. The performance of the bitmap manipulation instructions depends on the memory bus speed. The microoperation loop in the BVMAP instruction consists of three steps; the microoperation loops in the BVCPY and BVPAT instructions consist of two steps. All the microinstruction steps of the loops in the bitmap instructions include memory access operations. The memory access operation of the Gmicro/100 needs at least two clock cycles for its operation. So, the peak performance of the BVCPY and BVPAT instructions is 267 Mbits/s at a 33.3-MHz clock rate. The BVMAP instruction operates at 178 Mbits/s.

Effect of the prejump mechanism

A simulator evaluates the design decision and the performance of the Gmicro/100. The simulator is a register-transfer-level simulator written in PL/I. The microprogram and the programmable logic array (PLA) patterns used for the simulator are the same microprogram and the same PLA patterns used for logic simulation and the actual chip. The simulator has several mode switches that change the function of some hardware blocks. The simulator simulates the same hardware model to the Gmicro/100 and other several models depending on the mode switches.

Table 7 summarizes the benchmark programs used for the performance evaluation.[12,14] These benchmarks are written in the C language. The Dhry2 is the same C source code as the Dhry1, and the compiler in use offers increased execution time optimization compared to other compilers. We changed the execution step counts of all benchmark programs to make

Table 6. Bitmap instruction execution steps.

Instruction	Execution steps
BVMAP	If the source offset = the destination offset $3n + 15 + a + b + e$ Else $3n + 17 + a + b + c + d + e$
BVCPY	If the source offset = the destination offset $2n + 17 + a + b + e$ Else $2n + 17 + a + b + c + d + e$
BVPAT	$2n + 9 + a + b$

- n Number of 32-bit words on which the destination exists
- a If the destination block fetched first is not 32 bits long, a equals 2. Otherwise, a equals 0.
- b If the destination block fetched last is not 32 bits long, b equals 2. Otherwise, b equals 0.
- c If the source block fetched first is projected across the word boundary onto the destination area, c equals 2. Otherwise, c equals 0.
- d If the source block fetched last is projected across the word boundary onto the destination area, d equals 2. Otherwise, d equals 0.
- e If the source and the destination bits are processed backward from the tails to the heads, e equals 2. Otherwise, e equals 0.

Table 7. Summary of benchmark programs used for the performance evaluation.

Abbreviation	Full name of benchmark
Dhry1	Dhrystone Version 1.1
Dhry2	Dhrystone Version 1.1 (with a different compiler)
Quick	Quicksort
Sieve	The sieve of Eratosthenes
Fib1	Fibonacci series (with recursive function calls)
Fib2	Fibonacci series (without recursive function calls)
Ack	Ackermann function

Table 8. Runtime frequency of jump instruction.

Instruction	Frequency (percentage) Dhry1	Dhry2	Qsort	Sieve	Fib1	Fib2	Ack	Avg.
BRA	3.0	3.5	4.1	3.0	3.0	1.9	3.5	3.1
BSR	4.7	5.3	2.4	0.0	6.3	1.7	5.4	3.7
BCC								
Successful	4.4	3.5	10.0	19.5	3.4	15.7	6.0	8.9
Unsuccessful	4.1	5.7	10.2	6.1	3.0	1.9	3.5	4.9
Total	8.5	9.2	20.2	25.6	6.4	17.6	9.5	13.9
JMP	0.0	0.4	0.0	0.0	0.0	0.0	0.0	0.1
RTS	3.6	5.3	1.4	0.0	6.3	0.0	0.0	2.4
EXITD	1.1	0.0	1.0	0.0	0.0	1.7	5.4	1.3
Total	20.8	23.8	29.0	28.6	22.0	23.0	23.9	24.4
Total*	16.7	18.1	18.8	22.4	18.9	21.1	20.4	19.5

* Total runtime frequency of the jump instructions except for the unsuccessful BCC.

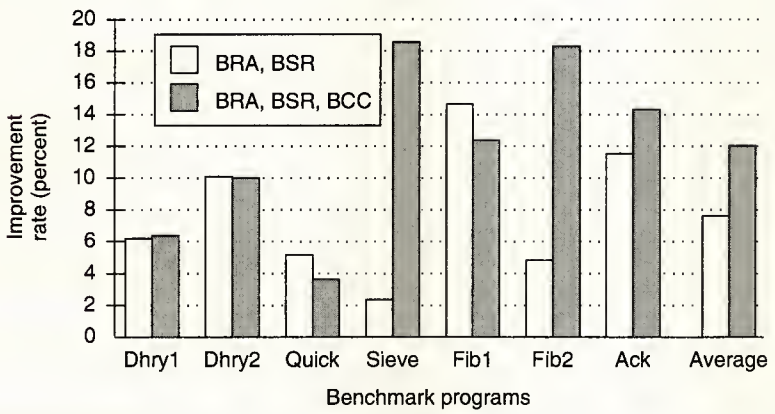


Figure 9. Prebranch effectiveness based on benchmark programs.

them sufficiently small to perform the simulations. We tuned each benchmark program to achieve about 5,000 clock cycles.

The simulator for each benchmark program counts the runtime frequency of jump instructions (see Table 8). In benchmark programs, only one JMP is used to execute memory indirect jump operations; the Gmicro/100 does not take a prejump for this instruction. The EXITD instruction both restores the registers and returns from the subroutine. Jump instructions constitute 24.4 percent of the instructions executed for the benchmark programs by frequency. Jump instructions, except the unsuccessful conditional branch instruction, constitute 19.5 percent of the instructions executed.

Effectiveness of the prebranch. Figure 9 shows the effectiveness of the prejump mechanism for unconditional branch instructions only and for both unconditional and conditional branch instructions. This figure charts the performance improvement rate of the prebranch. We calculate this rate by Rate = (Cnpb – Cpb) / Cpb, where Cpb is the number of clock cycles executed with the prebranch for the unconditional and conditional branch instructions. Cnpb is the number of clock cycles executed without the prebranch. We count the number of clock cycles executed from the enabling of the instruction cache and the accessing of the off-chip memory without waiting cycles.

Figure 10 shows that the Gmicro/100 executes the benchmark programs an average of 12 percent faster by taking the prebranch for unconditional and conditional branch instructions. The performance is not always increased by taking the prebranch for conditional branch instructions. In the cases of Quicksort and Fibonacci series benchmarks, the clock cycles increase by taking the prebranch for conditional branch instructions because the correct prediction rate of these two benchmark programs is low. Table 9 lists the probabilities of a branch being taken or not taken for conditional branch instructions and the probability of a correct prediction in a benchmark program.

Effectiveness of the prereturn. The probability of a prereturn being correct is almost 100 percent. Only one prereturn is not correct because of the ninth nest level subroutine of the Ack benchmark.

Figure 10 shows the effectiveness of the prejump mechanism for return-from-subroutine instructions with the runtime frequency of these instructions. The performance improvement rate of the prereturn is shown. We calculate this rate by Rate = (Cnpr – Cpr) / Cpr, where Cpr signifies the number of clock cycles with the prereturn and Cnpr is the number of clock cycles executed without the prereturn. We count the number of clock cycles from the enabling of the instruction cache and the accessing of the off-chip memory without waiting cycles.

Figure 10 also shows that the effectiveness of the prereturn depends on the runtime frequency of return-from-subrou-

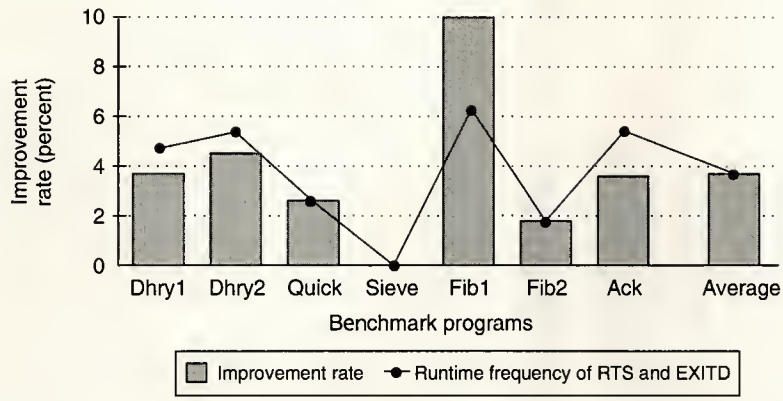


Figure 10. Prereturn effectiveness for return-for-subroutine instructions depends on their runtime frequency.

Table 9. Bcc behavior.

Behavior	Benchmark percentages Dhry1	Dhry2	Quick	Sieve	Fib1	Fib2	Ack	Avg.
Runtime frequency	8.5	9.2	20.2	25.6	6.4	17.6	9.5	13.9
Rate of successful branch of BCC (*b*)	51.6	38.5	49.3	76.0	52.4	89.1	63.2	60.0
Correct rate								
Overall prediction	67.7	76.9	56.2	74.7	38.1	78.2	73.5	66.5
Prediction for a successful branch (*p*)	68.8	70.0	51.4	83.0	36.4	87.8	71.9	68.1
Prediction for a unsuccessful branch (*q*)	66.7	81.3	60.8	48.1	40.0	0.0	64.0	51.6

tine instructions. The Gmicro/100 executes the benchmark programs an average of 3.7 percent faster with the prereturn than without it.

Effectiveness of the instruction cache for the prejump. The instruction cache is very important in helping the prejump mechanism's effectiveness in restoring performance lost by jump instructions. When the prejump is taken in the D stage without the instruction cache, the instruction fetch from the new path sometimes conflicts with the operand access in another stage of the pipeline. The priority of the instruction fetch in the D stage is the lowest. If the memory bus is used for operand access, the D stage waits until the operand access ends.

Figure 11 also shows the effectiveness of the prejump for all jump instructions and the total performance improvement rate of the prejump mechanism. We calculate this rate by Rate = (Cnp – Cp) / Cp, where Cp is the number of execution clock cycles with the prejump and Cnp is the number of execution clock cycles without the prejump. We count the number of execution clock cycles from when the instruction cache is on or off and from the accessing the off-chip memory with either zero or two waiting cycles.

The Gmicro/100 executes the benchmark programs an average of 16.8 percent faster by using the prejump mechanism when the cache is on and the off-chip memory is accessed without waiting cycles. Prejump effectiveness with the instruction cache off is less than that with the instruction cache on for all benchmark programs. The reason is that the timing of the instruction fetch from the jump target is critical for the performance. The prejump's effectiveness decreases when two-wait-state memory is used instead of zero-wait-state memory. Prejump effectiveness is especially minimal when the instruction cache is off. If the utilization rate of the processor bus is too high to fetch the instructions at the prejump target address, the prejump mechanism cannot increase the performance. The instruction cache helps the prejump mechanism to increase the performance of the Gmicro/100 when the external memory is slow.

Effect of bitmap-manipulation instructions

We evaluated the performance of the bitmap instructions by executing primitive bitmap-manipulation operations. The performance of bitmap instructions compares with the performance of loop programs that consist of simple instructions.[15]

Figure 12 diagrams the result of the comparison between the BVMAP instruction and the software loop of the same function. This figure shows that the offsets of the source- and destination-bit strings are different. The bitmap instructions process much faster in this case. The BVMAP instruction is 4.5 times faster than the software loop.

Even when the offsets of the source and destination bit strings are the same, the bitmap instructions process faster than the software loops. Comparisons between BVMAP and the software loop and BVCPY and the software loop appear in Figure 13a and b on p. 71. When the offsets of the bit strings are the same, the bit strings are processed without shift

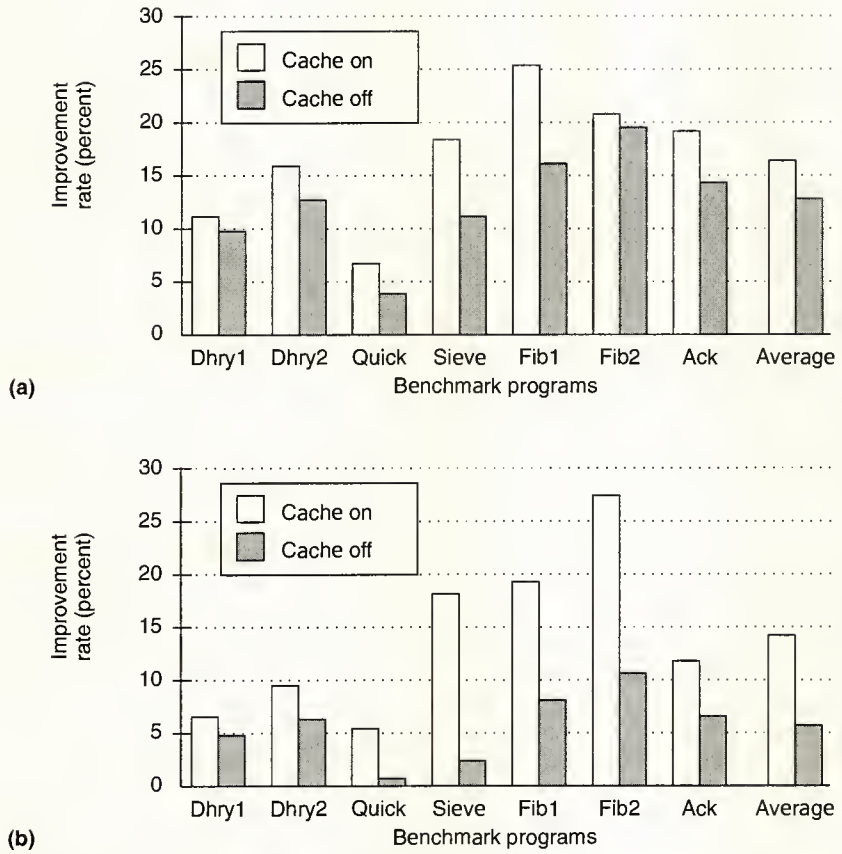


Figure 11. Prejump effectiveness for all jump instructions: without wait cycles (a) and two wait cycles (b).

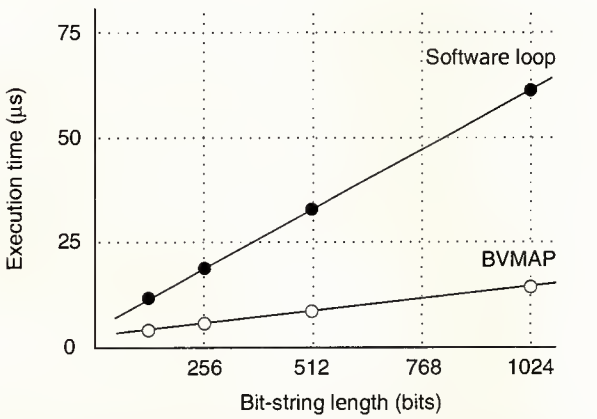


Figure 12. Performance comparison between the bitmap instruction and software loop (source offset = destination offset)

operation. So the performance difference is less than if the offsets are different. But the bitmap instructions are still two to three times faster than the software loop.

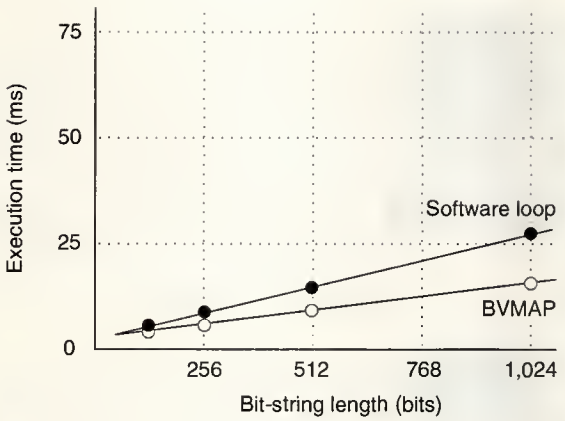
The BVCPY instruction is useful not only for the bit string copy operation but also for other data copy operations in the memory. All BVCPY operations are memory-to-memory. If the data is not word-aligned (for example, character strings are not always word-aligned), the BVCPY instruction processes much faster than the software loop because the BVCPY instruction accesses the memory by the word block in all cases.

ASSP approach

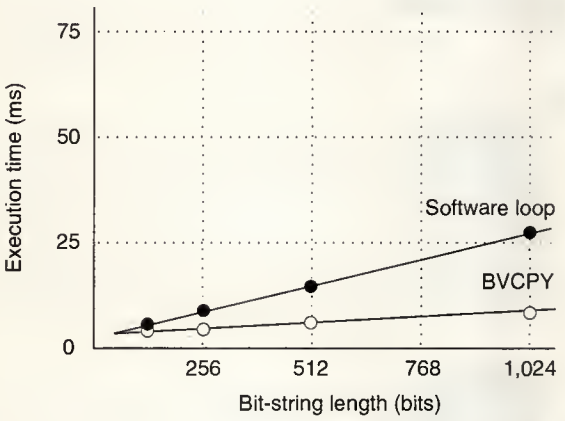
The Gmicro/100 ASSP is the first application-specific standard product (ASSP), the core microprocessor unit of which is the 1.0-µm version. We developed the ASSP to demonstrate the Gmicro/100 ASIC (application-specific integrated circuit) development. Our ASIC approach provides users with the performance of the 32-bit microprocessor, availability of powerful software development tools, and system integration features. Peripheral functions that are useful for personal-use information equipment (such as a personal computer based on BTRON, or business version of the TRON specifications) will be integrated into one VLSI chip with the Gmicro/100.

The Gmicro/100 ASSP includes an interrupt controller, a bus arbiter, timers, a DRAM refresh controller, a wait controller, a DMA controller, and an universal asynchronous receiver transmitter (see box). Using a 1.0-µm, double-metal CMOS technology, the ASSP integrates 420,000 transistors in die size of 12.78 × 14.68 mm^2. The core MPU uses 340,000 transistors (77 percent of whole transistors), and peripheral functions use an additional 80,000 transistors (23 percent). Figure 14 is a photograph of the chip.

The ASSP's hierarchical design uses a cell-based design approach. To shorten design time, designers may use a standard cell approach, which lays out the macro cells of the chip, except for the core MPU. To test the core MPU easily, designers use test circuits to apply test vectors for the core MPU from the external pins. The chip is housed in a 179-PGA (pin grid array) package. For surface mounting in higher density and cost reduction, we are developing a 208-pin quad flat package (QFP). Figure 15 shows the chip block diagram.



(a)



(b)

Figure 13. Performance comparison: BVMAP versus software loop (a) and BVCPY versus software loop (b).

WE DISCUSSED THE PIPELINE FEATURE, prejump mechanism, bitmap manipulation instructions, and the ASSP approach of the Gmicro/100. We have developed personal-use information equipment and embedded controllers, and we use the first-stage Gmicro series processors—the Gmicro/200 and /300—in applications such as factory automation controllers and communication processors.

Currently, we are developing the second-stage Gmicro series processors, which include floating-point operation units and are much faster than the first-stage processors. We are also developing new dedicated VLSI chips that boost performance of the second-stage processors.

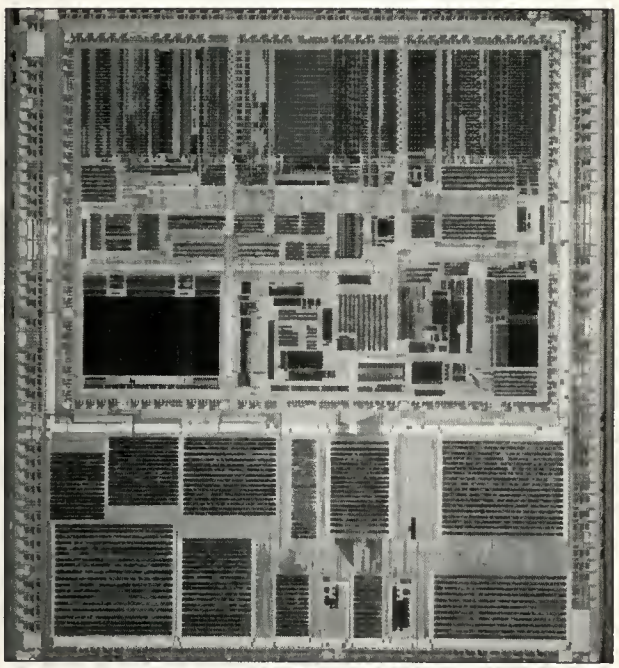


Figure 14. Gmicro/100 ASSP.

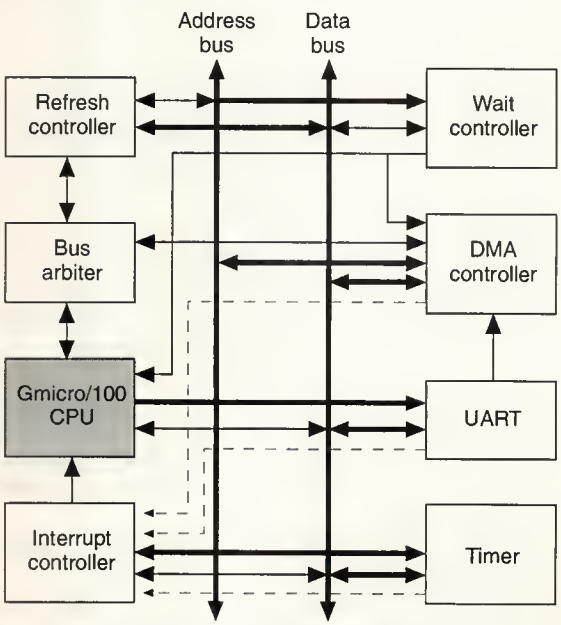


Figure 15. Diagram of the Gmicro/100 ASSP.

Acknowledgments

The authors acknowledge Ken Sakamura and the Gmicro team members for their helpful discussions. We also thank all of the engineers on the TRON project for their useful comments.

References

1. K. Sakamura, "Architecture of the TRON VLSI CPU," *IEEE Micro*, Vol. 7, No. 2, Apr. 1987, pp. 17-31.
2. J.K.F. Lee and A.J. Smith, "Branch Prediction Strategies and Branch Target Buffer Design," *Computer*, Vol. 17, No. 1, Jan. 1984, pp. 6-22.
3. T. Shimizu et al., "A 32-Bit Microprocessor with High Performance Bitmap Manipulation Instructions," *Proc. IEEE Int'l. Conf. Computer Design*, IEEE Computer Society Press, Los Alamitos, Calif., Oct. 1989, pp. 406-409.
4. T. Shimizu et al., "A 32-bit Microprocessor Based on the TRON Architecture: Design of the Gmicro/100," *Proc. 33rd IEEE Computer Society Int'l. Conf.*, IEEE CS Press, Feb. 1988, pp. 30-33.
5. H. Inayoshi et al., "Realization of the Gmicro/200," *IEEE Micro*, Vol. 8, No. 2, Apr 1988, pp. 12-21.
6. T. Kitahara and T. Satoh, "The Gmicro/300 32-bit Microprocessor," *IEEE Micro*, Vol. 10, No. 3, June 1990, pp. 68-75.
7. T. Yoshida et al., "A Strategy for Avoiding Pipeline Interlock Delays in a Microprocessor," *Proc. IEEE Int'l. Conf. Computer Design*, IEEE CS Press, Sept. 1990, pp. 14-19.
8. D.J. Lilja, "Reducing the Branch Penalty in Pipelined Processors," *Computer*, Vol. 21, No. 7, July 1988, pp. 47-55.
9. S. McFarling and J. Hennessy, "Reducing the Cost of Branches," *Proc. 13th Ann. Int'l. Symp. Computer Architecture*, IEEE CS Press, June 1986, pp. 396-403.
10. D.R. Ditzel and H.R. McLellan, "Branch Folding in the CRISP Microprocessor: Reducing the Branch Delay to Zero," *Proc. 14th Symp. Computer Architecture*, IEEE CS Press, June 1987, pp. 2-9.
11. The TRON Association, "Specification of the Chip Based on the TRON Architecture," TRON Association, Tokyo, 1990.
12. R.P. Weiker, "Dhrystone: A Synthetic Systems Programming Benchmark," *Comm. ACM*, Vol. 27, No. 10, Oct. 1984, pp. 1013-1030.
13. C. Franklin and C. Rosenberg, "Inline Procedures Boost Performance on TRON Architecture," *TRON Project 1990*, Springer-Verlag, 1990, pp. 275-292.
14. Editorial Staff, "High-Tech Horsepower," *Byte*, Vol. 12, No. 8, July 1987, pp. 101-108.
15. M. Sakamoto, T. Shimizu, and K. Saitoh, "The Evaluation of M32/100's Bitmap Instructions Used in the Graphic Primitive," *TRON Project 1990*, Springer-Verlag, 1990, pp. 261-271.

Toyohiko Yoshida is a researcher with the Advanced Microprocessor Development Group at Mitsubishi Electric's LSI Laboratory. His interests include microprocessor research and design.

Yoshida has a BE and ME in electronics engineering from Kyoto University, Japan. He is a member of the IEEE and the Institute of Electrical and Communications Engineers of Japan (IECEJ).

Toru Shimizu is a researcher in the same Mitsubishi group. His interests include microprocessor architecture and software.

Shimizu has a BS, MS, and PhD in computer science from the University of Tokyo. He is a member of the IEEE, the Association for Computing Machinery, IECEJ, and the Information Processing Society of Japan (IPSJ).

Shigeo Mizugaki is an engineer in the Mitsubishi's Microcomputer Department at Kita-Itami Works in Itami, Japan. His interests include design of 8- and 16-bit microcontrollers and 32-bit microprocessors. Mizugaki has a BE and ME in electronics engineering from Kyoto University.

Junichi Hinata is a manager of the Advanced Microprocessor Development Group. His interests include the development of small computers, VLSI custom processors, and 32-bit microprocessors.

Hinata has a BE in electronics engineering from Hokkaido University, Sapporo, Japan. He is a member of the IPSJ.

Address questions concerning this article to Toyohiko Yoshida, LSI Device Development (III), First Group, Mitsubishi Electric Corp.LSI Laboratory, 4-1, Itami, Hyogo 664, Japan; or e-mail at yoshida@micro.1si.melco.co.jp.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 156 Medium 157 High 158

Sure System 2000

continued from p. 31

fers at high speed and efficiency increases in maximizing shared-storage bus use.

Address space. Figure 4 illustrates the address space allocated to each CPU in each processor module. The CPU has a 4-Gbyte address space. The first 2-Gbyte half, beginning with address 0, is mapped to the shared-storage module space (address space for shared storage). The second half is used for the individual processor module, and it is defined as a space for handling the local storage units in the processor module and external registers.

The SS2000 can contain multiple sets of paired physical shared-storage units. Each module connected to either the shared-storage or I/O bus uses an ID on the bus for bus operations involved in different modules. In program mode, each processor module uses an independent table for converting addresses in shared-storage module space to IDs on the bus. These tables function independently of the duplicate shared-storage buses. If an attempt is made to access shared-storage module space, the processor module hardware references the conversion table to generate the ID on the bus and the relative address. Then it sends a command for shared-storage module access to the shared-storage bus. The operating system initializes conversion tables before shared-storage module access.

In DMA mode, address space is defined totally independently of CPU address space. The operating system directly specifies the 8-bit shared-storage module ID on the bus and the 32-bit relative address within the physical shared-storage module determined by that ID, as stated earlier. In other words, shared-storage module address space in DMA mode ensures the allocation of a very large space (1 terabyte) represented by 4 Gbytes multiplied by the ID on the bus. In program mode, address space in the shared-storage module is limited to 2 Gbytes, while in DMA mode, address space is expanded to 1 terabyte.

Semaphore information. Multiprocessor instructions received from CPU instructions place semaphore information on the shared-storage module in what is considered to be a special program mode. When a multiprocessor instruction is executed in the module space, data read from this space is compared with a reference value. When the two agree, replacement data can be written in the module. This module is locked from the time reading begins to the time writing ends, or a mismatch is detected.

Dual writing. Since the module is duplicated to make it fault-resistant, the operating system writes the same data to each module. To reduce operating system overhead, the module contains a hardware facility for dual writing independent of the access mode. For reliability, the hardware first attempts to write to one shared-storage module; when writing proves successful, it writes to the other module in the pair.

Though the operating system specifies the write sequence in program mode, the sequence can be changed dynamically. The conversion table described earlier holds the information to support decisions about the write sequence and single or dual writing. In DMA mode, the access control word specifies the sequence of each operation. In dual-writing mode, access for a read is made to the module specified for the first write.

If an attempt to write to a shared-storage module fails, the system provides a facility for notifying the operating system of the unsuccessfully accessed module. This facility recovers the contents of the module by referencing the access sequence.

Access path. Each module has duplicate shared-storage buses. For ordinary operation, either bus can be used. In program mode, the operating system can specify the access path, which can also be changed dynamically.

In DMA mode, multiple DMA channels exist for each path. When the operating sys-

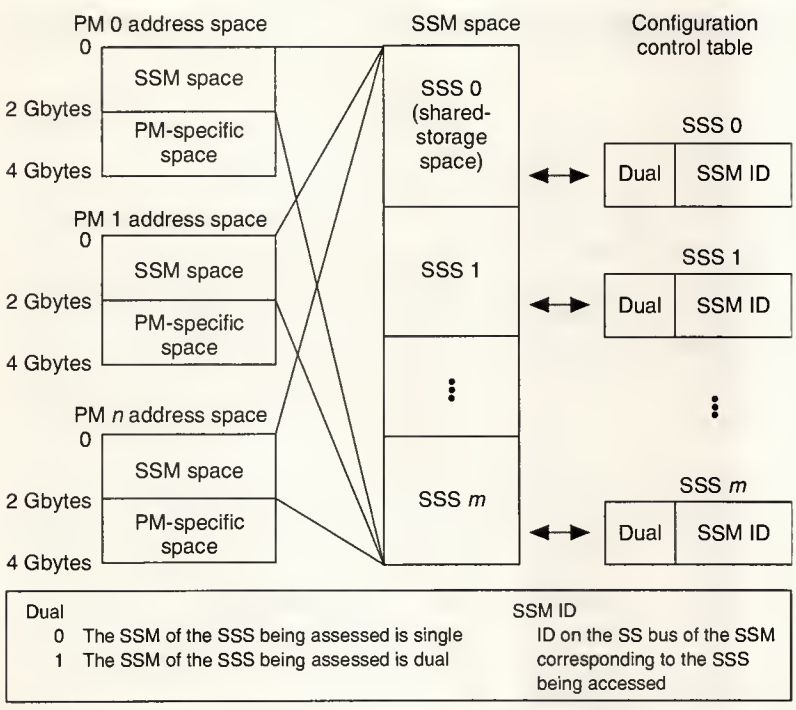


Figure 4. Address space.

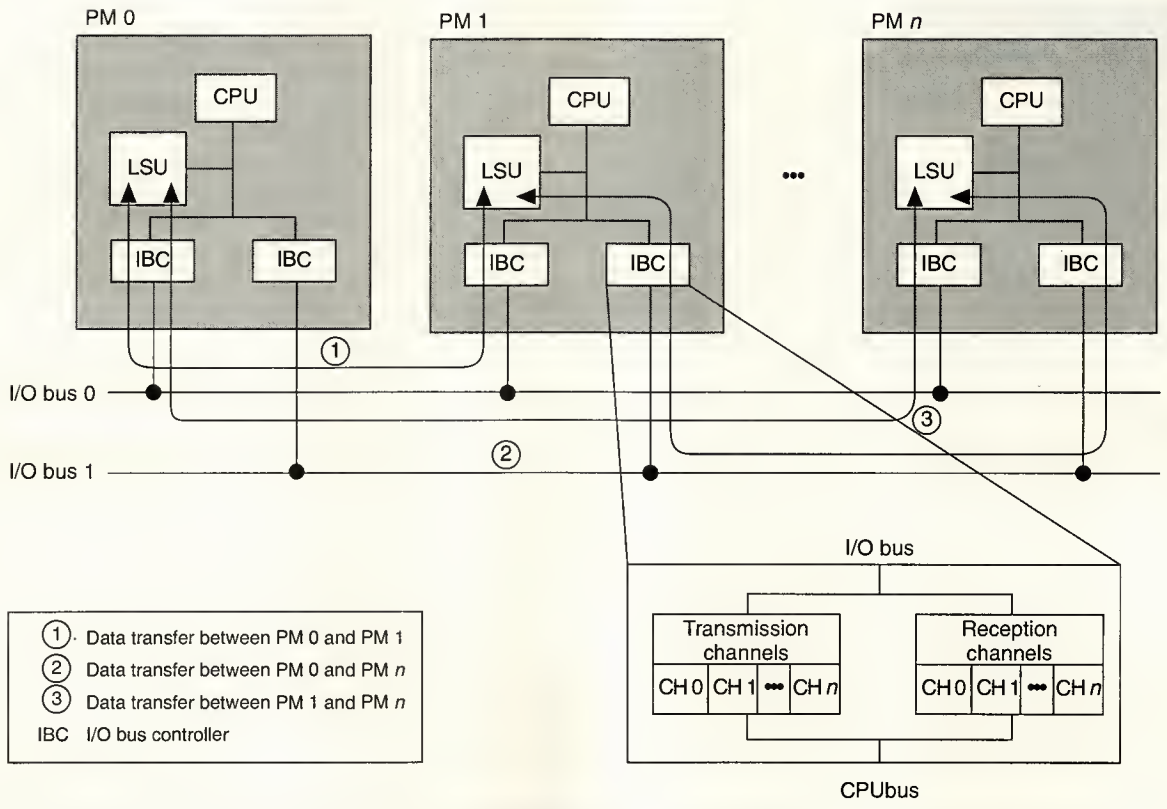


Figure 5. Data transfers among processor modules.

tem selects a channel to be activated, the access path is determined automatically. Multiple paths can be activated at the same time in this mode.

Dual writing and access-path selection immunize the module against faults and load distribution problems. If a fault occurs in the shared-storage module or shared-storage bus, the operating system manipulates configuration information found on each processor module for use in accessing the module so that functionally reduced running can begin quickly. For instance, if a shared-storage module fault occurs, the dual-writing mode is canceled; if an shared-storage bus fault occurs, the access path is manipulated, enabling functionally reduced running.

Storage is usually heavily loaded, and the shared-storage module in the SS2000 is not free from this problem and, in fact, is to be expected. However, the operating system can distribute the module's load by modifying the specification execution sequence and distribute the shared-storage bus load by assigning access paths to processor modules.

Data transfers among modules. The interprocessor direct data transfer facility is a means for multiprocessor coupling. Any processor module can send data directly to or receive it from any other processor module connected to the I/O bus. More accurately, data transfers among local storage units in processor modules without regard to the instruction running on the CPU (Figure 5).

Each processor module controls data transfers through multiple transmission and reception channels, which transfer data to other processor modules. These channels can be operated independently, with simultaneous transfers to or from multiple processor modules. Data transfers in one of two modes: short mode when transferring data of one page or less, and long mode when transferring data exceeding one page at one time. Every transmission and reception channel can be dynamically switched between these two modes.

Short mode. A control word in the transmission channel and the reception channel controls each channel. The operating system provides the transmission channel control word and data buffer on the local storage unit, then activates the channel to start transferring data. It also provides the reception channel control word and data buffer, then activates this channel to ready the processor module for reception. In general, the reception channel of each processor module is ready to receive data in this mode.

Data transmitted in short mode arrives at one of the reception channels that ready for reception. The control words

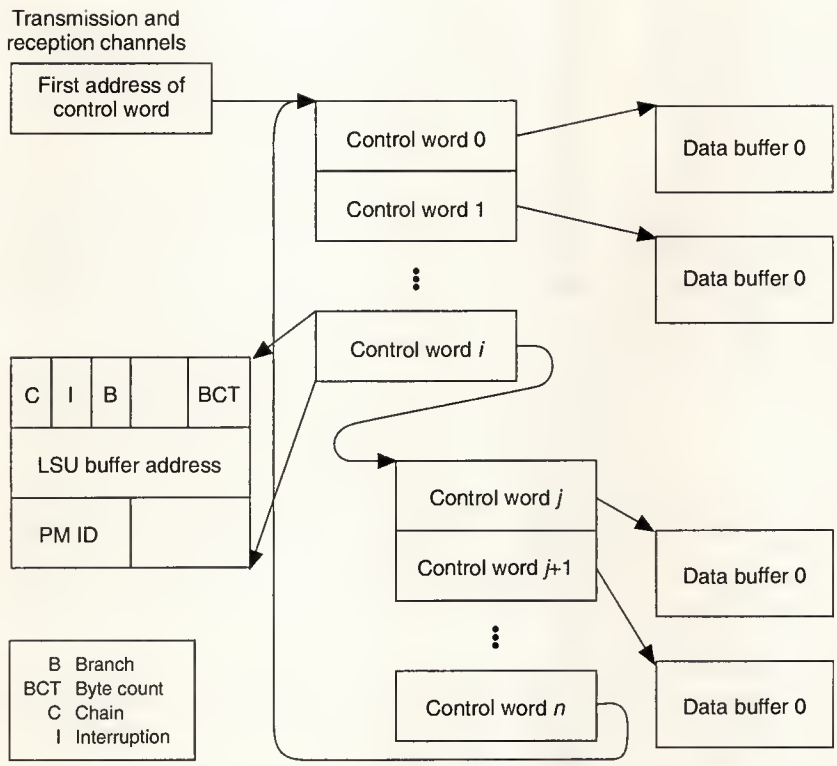


Figure 6. An Interprocessor-module transfer operation. The processor module ID of the reception destination is specified on the transmission channel, while the transmission source is specified on the reception channel.

contain branching and external interrupt-generating facilities in addition to chaining facilities. These facilities enable the buffer to be used efficiently under operating system control (see Figure 6).

A reception channel's chained control words can hold received data from different processor modules, and a transmission channel's chained control words can transmit data to different modules. The channels send an external interrupt request to the CPU of the module when the chain is cut in response to the termination of transmission or reception or when the control word specifies generation of an external interrupt request.

Long mode. In this mode data transfer begins after particular processor modules negotiate with each other to acquire data buffers. Data can be transferred without any size restriction. This mode is identical to short mode in control mode chaining, branching, and external interrupt requesting. However, data indicated by all control words is transmitted to the same module. For example, this mode enables a large amount of transmission and reception data to be transferred directly to and from the line.

I/O architecture

The I/O bus connects SS2000 I/O control modules to the processor modules. Each processor module can directly access I/O modules on an equal footing.

SS2000 I/O control modules manage the physical device control tables that correspond to their subordinate physical devices (disks or lines) and the logical device control tables that correspond to individual processor modules. When a processor module issues an I/O control request, the I/O control module places the control information into the corresponding logical table. Queuing for this table is available for the service. When the physical device becomes available, actual I/O accesses are run, after coordinating the logical table with the physical table.

A CPU instruction starting an I/O operation is free from interlocking with the I/O device and I/O control module, unlike that which occurs in conventional operations. The start of I/O processing is complete when the appropriate information is placed in the buffer of the module. Therefore, it is not influenced by a wait period for I/O bus acquisition or by operation within the I/O control module. It takes place as fast as a general-purpose memory access.

Each I/O control module contains facilities for queuing I/O processing requests by processor modules. For example, suppose that a particular module requests a particular I/O device for processing and that the I/O control module is servicing this request. When another module issues an I/O request to the same I/O device, the request is not rejected because of a busy condition, but is put into the queue within the I/O control module.

Because of this, for direct access from each processor module and queuing of I/O processing requests, the operating system of each module is free to access each I/O device directly without concern for the I/O processing condition of other modules (see Figure 7 on the next page).

The SS2000 provides especially reliable data transfer for the disk on which user data is stored. Data writes to the disk via the I/O control module between storage and the disk. In general, data moves from storage to I/O control modules and from I/O control modules to disk in parallel via a FIFO (first in, first out) buffer within the I/O control module. One problem with this scheme is that the I/O control module may

become unable to obtain write data during transfer because of a fault in the processor module or I/O bus. In such cases, the system pads the disk, causing some disk contents that are entirely old, entirely new, or in an intermediate state—neither old nor new.

The SS2000 supports a special disk writing mode in which store data is placed in the SCSI module, the I/O control module for the disk, and then is transferred to the disk. In other words, transfers occur in two steps: one from the local storage unit of the module to the buffer in the SCSI module and the other from the buffer to the disk. The second step does not start unless the first step ends normally. If the processor module becomes faulty during disk writing, it is assured that disk data is entirely old or entirely new.

SXO

Finally, let us describe the software architecture of the SXO Sure System 2000 expandable operating system,[9,10] depending on the hardware architecture just mentioned.

Objectives. One objective of the SS2000 includes fault tolerance and continuation of system operation in the presence of component failure. The system must be robust enough to withstand hardware failures and software system component failure. Another requirement is continued system operation even when its software is being fixed, or its hardware is being replaced, upgraded, or added to the system. Designers kept these objectives in mind when designing the SXO.

Architecture. SXO has a kernel and many server processes, which offer total service to application programs. The kernel resides in every processor module. A capability scheme controls resource management (including messages), and SXO supports multiple virtual memory spaces. Each server resides in its own virtual address space.

The interface between virtual address spaces are based on message passing. A capability scheme controls the resource management (including the message).

SXO is divided into different layers:

- **Kernel.** Basic functions—for message passing, virtual memory management, and I/O control—reside in the same address in each virtual memory address space of every processor module.
- **Global service.** Servers that reside in one processor module (and in another backup module) offer operating system services such as a file operating system control.
- **Local service.** A set of subroutines resides in the processor modules to offer services that do not need global service (that is, services that can be finished in one virtual address space). This layer also offers the interface between application and servers.

Servers. In a typical configuration, SXO has a few dozen servers running. Each server has one current virtual space and a backup server on a different processor module. When an error in the current address space is detected, the spare address space appears on line to replace the faulty address space. The use of multiple virtual address spaces helps the system isolate a software fault in a server's address space. Seen from the view of fault tolerance, message passing helps the designer of an operating system to increase the independence of each server. The only interface between different servers are these messages.

Traditional mainframe systems often achieve fault tolerance by using hot/standby systems; that is, standby systems exactly replicating the computer system are ready to take over a faulty, or hot, system. It usually takes a few minutes to boot the backup system once SXO detects a failure in the primary system.

SXO replaces only the operating system server fault with the spare address space. Hence, the switch takes place in a few seconds as opposed to the few minutes required in typical hot/standby systems.

Application Program Interface. The API in SXO is based on the Posix standard.[11] Simple user applications don't have to pay much attention to the underlying system. In this way, typical Unix application programs such as Emacs have been ported to SXO.

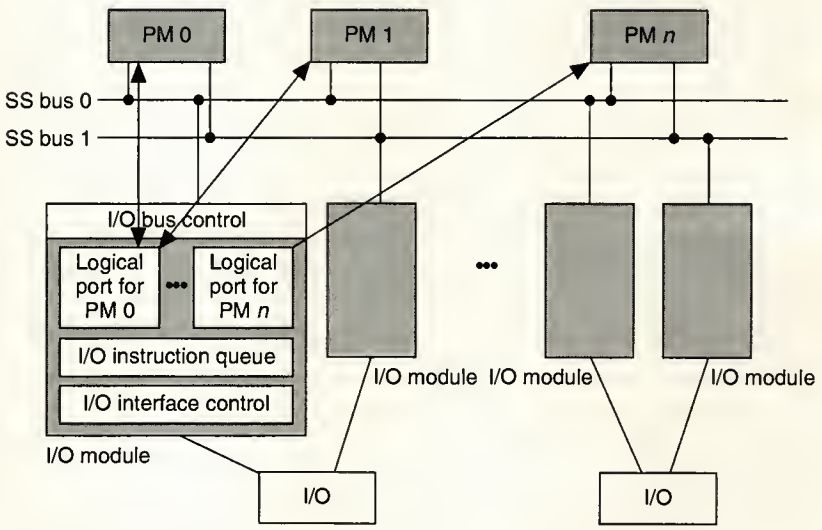


Figure 7. I/O architecture.

The user program doesn't have to concern itself with the message-based interface. The local service residing in the same processor module translates the API into the message-based ones. And the kernel in the same processor modules then talks to the appropriate server and returns the result to the application in a form expected by the Posix-based API. So programming applications are easy.

Most of the system calls supplied in the API set are atomic. The operation either takes place completely or nothing is done at all. For example, say an application tries to create a file. Before the system call to do so fails due to faulty hardware or software, SXO makes sure that it replaces the faulty server. The system service call succeeds completely when the system call operation finishes, and the result is returned to the original application. SXO does not inform the application that fault-tolerant processing is taking place behind the scenes.

This atomicity is important to make sure that the system as seen from the user remains in a consistent state. Also, by offering atomic system calls, the user can write rather clean code without cluttering it up with retries and other recovery/clean-up operations. Atomicity saves coding time for application writers.

Take over. Most of the servers in SXO have a current virtual address space and a spare virtual address space. They reside in different processor modules to increase fault tolerance and make it possible to fix the server programs dynamically.

When an error in the current space is detected, the kernel aborts this space and notifies all other servers to release the resources held by the dying server. In parallel, the kernel notifies the spare space to begin processing as the current space of the server.

When a corrected version of the software (periodic fix or urgent fix) is being installed, the following steps take place. First, SXO kills the spare space with the current version of the software, then it starts a new spare space with the corrected version. It kills the current space of the server, which is taken over by the new spare space with the corrected software.

When a running server is shut down and taken over by another space of the server, information must be passed to the new address space so that pending operations can be finished to give applications atomic services. A shared-storage module passes the information in this case at a highly logical description of what actions were being performed. Detailed information can't be passed at a physical level—such as an entire memory page image to be used for I/O and so on. We want to minimize the amount of passed information. Also, since corrupted memory causes many operating system module failures, we don't want to force the new running server to access the same corrupted memory by passing the detailed internal state of the original failed server.

The use of SSM speeds up the operation for the backup server to take over. Contrary to some fault-tolerant systems in which the information to back up a software module is sent by synchronous communication, SXO makes it possible to kill the backup module and restart it with a version of the software dynamically by passing the information in SSM.

Results. By adopting the software architecture just described, SXO now allows users to dynamically

- replace a server process,
- replace a hardware component,
- upgrade the operating system software, and
- add a hardware component.

DESIGNERS OF THE SURE SYSTEM 2000 developed it on the following principles:

1) processors access shared memory on a uniform basis,
2) data transfers directly between processors,
3) individual processors can access any I/O device in the system,
4) the operating system (SXO) can be fixed and the hardware module replaced, and
5) the hardware or the software can be upgraded or added dynamically.

Employing these architectural principles lets the SS2000 achieve an excellent fault-tolerant system. Both hardware and software (SXO) modules have local redundancy, so that no one fault can cause a system crash. In addition, the SS2000 achieves the new type of multiprocessor architecture with high reliability and a high degree of processor coupling. The SS2000 provides a highly adaptable computer environment and 24-hour-a-day, nonstop operation.

References

1. J.N. Gray, "Why Do Computers Stop and What Can Be Done About It?," Tech. Report 1985.7, Tandem Computers, 1985.
2. O. Serlin, "Fault-Tolerant Systems in Commercial Application," *Computer*, Vol. 17, No. 8, 1984, pp. 19-30.
3. A. Avizienis, "On the Achievement of a Highly Dependable and Fault-Tolerant Air Traffic Control System," *Computer*, Vol. 20, No. 2, 1987, pp. 84-90.
4. T. Kitahara and T. Sato, "The Gmicro/300 32-Bit Microprocessor," *IEEE Micro*, Vol. 10, No. 3, Jun. 1990, pp. 68-75.
5. K. Sakamura, "Architecture of the TRON VLSI CPU," *IEEE Micro*, Vol. 7, No. 2, Apr. 1987, pp. 17-31.
6. K. Sakamura, "TRON VLSI CPU: Concepts and Architecture," *Proc. TRON Project*, Springer-Verlag, Tokyo, 1987, pp. 200-308.
7. N. Kurobane, T. Kato, and S. Tanaka, "A Fault-Tolerant Operating System Using Essential Recovery Data," *Proc. 40th Ann. Conf.*

Information Processing Society of Japan (in Japanese), 1990, pp. 750-751.
8. A. Tannenbaum and R. van Renesse, "Distributed Operating Systems," *ACM Computing Surveys,* Vol. 17, No. 4, 1985, pp. 419-470.
9. M. Date and H. Yoshida, "Operating System SXO for Continuously Operable Sure System 2000, Vol. 1, An Overview," *Proc. 42nd Ann. Convention IPS Japan,* 1991, pp. 77-78.
10. Y. Eto, M. Kasuga, and K. Nakamura, "Operating System SXO for Continuously Operable Sure System 2000, Vol. 2, The Architecture," *Proc. 42nd Ann. Convention IPS Japan,* 1991, pp. 79-80.
11. *ISO/IEC Std 9945-1: 1990 Information Technology, Posix* (formerly *IEEE Std 1003.1-1990*), IEEE Computer Society, Los Alamitos, Calif., catalog no. 1019.

Akira Kabemoto is a senior engineer in the Sure System 2000 Design Group, Main Frame Department, of Fujitsu Limited. He helped develop a communication control processor and has been engaged in the development and the architecture design of the Sure System 2000.

Kabemoto received the BE degree in electrical engineering from the Kyoto University.

Hiroshi Yoshida is a section manager in the SXO Development Group, Software Division, at Fujitsu Limited. He developed the SXO, concentrating on the design of its file management system.

Yoshida received the BE degree in electronics engineering and the ME degree in information engineering from the University of Tokyo. He is a member of the IEEE and the IEEE Computer Society.

Direct questions concerning this article to Akira Kabemoto, First Design Section Computer 4th Engineering Dept., Main Frame Division, Fujitsu Limited, 1015, Kamikodanaka Nakaharaku, Kawasaki 211, Japan.

ITRON-MP

continued from p. 27

A kernel resource implemented on private hardware resources of processor p can be accessed from other processors by requesting p for the access. In this case, the kernel resource is accessible from other processors, though it is realized on private hardware resources. This is an example of the disagreement. This implementation method is common in loosely coupled multiprocessor systems. In shared-memory multiprocessor systems, however, this method is inefficient for implementing all system calls, and it is not regarded as a standard implementation method for ITRON-MP.

We list other examples in which the class of a kernel resource does not agree with that of the hardware resource on which the kernel resource is realized:

- When the access cost of a local hardware resource is almost the same as that of a private hardware resource, we can implement a private kernel resource using a local hardware resource.
- When little difference occurs in the system between the access efficiency of local hardware and that of global hardware resources, it is not necessary to divide the class of local and global kernel resources from the standpoint of access efficiency.
- When a kernel resource is realized using hardware resources belonging to different classes in spite of the principle mentioned previously, the kernel resource should be reclassified as a new class. Suppose we implement a memory pool using two memory boards. The memory pool becomes unusable when one of the memory boards is out of order. Therefore, from the standpoint of replaceable units, the memory pool is reclassified to a new class with such a characteristic.
- When on-line maintenance in the system is not possible, kernel resources need not be classified from the standpoint of replaceable units.

Classification of tasks. We use an active hardware resource (a processor) and passive hardware resources (memory units and I/O devices) to implement tasks. We call this kind of kernel resource an active kernel resource; its classification differs from other kernel resources. Exception handlers and the extended system calls of ITRON-MP are other examples of active kernel resources. We describe only the classification of tasks here, because other active kernel resources are classified similarly.

A task requires regions for its program code, data, stack, task control block (TCB), and so on. In this case, we found that the principle that a kernel resource should be realized using a class of hardware resources is too restrictive. In ITRON-

MP, the kernel manages only the stack region and the TCB of a task; the linker or loader manages the other regions. Moreover, in typical implementation, the accessibility of a task agrees with the accessibility of its TCB. (Accessing a task includes creating, starting, suspending, and terminating the task, as well as getting the task's status.) Accordingly, the class of the hardware resource on which its TCB resides classifies the task.

In spite of this relaxation of the principle, each region for a task should be located on the same replaceable unit when on-line maintenance is necessary. Also, the read-only regions of a task, the program code region and the constant data region, can be duplicated to the local memory of every processor that executes the task.

Tasks are further classified by their executability. A task can be executed on processor p if all hardware resources that constitute the task are accessible from p. When the runtime efficiency is important, however, the classification by its executability usually agrees with the classification by the access efficiency of the hardware resources. The architecture in the previous example greatly degrades the runtime efficiency to execute a task that has its program code region and data region residing on the memory connected to the local bus of processor p on another processor than p. Consequently, we make the task executable only on p in this case. It is also inefficient to execute a task located on the memory connected to the backplane bus only on one processor, because it wastes the bandwidth of the backplane bus.

As the high runtime efficiency is among the most important features of ITRON-MP, the classification of tasks by their executability agrees with that of hardware resources on which the tasks are located by their access efficiency. This simplification makes the classification of active kernel resources agree with that of other kernel resources, but causes an inconvenience in the following case. A task realized on shared hardware resources and executable on some processors for fault tolerance or load balancing is often bound to a processor for efficiency. We prepare a system call that temporarily restricts the set of processors executing the task for this case.

Resource accessibility from tasks. In previous sections, we saw that accessibility from processors classifies kernel resources. However, the accessibility of kernel resources from tasks is more important for programmers. In the architecture illustrated in Figure 2, global tasks are accessible and executable by all processors. Tasks accessible from all processors and executable by processor p are called $\{p\}$-local tasks. Tasks accessible and executable by processor p are called $\{p\}$-private tasks, by the principle that the classification of tasks by executability agrees with their location.

Table 1 lists the accessibility of kernel resources from these classes of tasks. In this table, $\{p\}$-private tasks can access shared resources, but cannot wait for them. The reason: If a $\{p\}$-private task t waits for a shared kernel resource such as a semaphore, another task executed on a processor other than p that tries to access the resource cannot wake up the task t since the task cannot access the TCB of t.

Table 1. Accessibility of kernel resources from tasks.

Resource	Tasks: Global	$\{p\}$-local	$\{p\}$-private
Shared	Accessible	Accessible	Only accesses without waiting are possible
$\{p\}$-private	Inaccessible	Accessible	Accessible
$\{q\}$-private	Inaccessible	Inaccessible	Inaccessible

In general, for any sets of processors S, T, and U, we call a kernel resource that can be accessed from the processors belonging to S as a S-shared resource. We call a task, which can be executed on processors belonging to T and accessed from the processors belonging to U, U-shared and T-bound. A U-shared, T-bound task can access an S-shared kernel resource, a resource if and only if $T \subseteq S \wedge S \subseteq U$.

Though private tasks of a processor may seem to be useless because of their very limited accessibility, they are very effective on some occasions.

As a microprocessor becomes less expensive, designers can usually relieve the main processor of a system from low-level I/O processing by attaching a dedicated processor to each I/O device. For example, many commercial disk controller boards contain a processor. This kind of intelligent I/O controller board usually has its own protocol to communicate with the main processor. On the other hand, the main processor should run the interface tasks that communicate with the boards. If we use the ITRON-MP kernel to implement I/O control software on the boards, we can use the software to exchange information with the main processor through the communication resources of ITRON-MP. This process is beneficial because of the standardization of the communication protocols between the main processor and I/O controllers, and because no overhead occurs for protocol conversion.

The tasks that control I/O devices and that are executed on I/O control processors need not communicate directly with tasks executed on other processors by using some dedicated tasks for the communication. Therefore, we can implement I/O controlling tasks as private tasks. The runtime efficiency of private tasks is high, because no mutual exclusion with other processors is necessary. Also, the implementation cost is low. Consequently, private tasks are worthwhile in spite of their limitations.

Kernel resource identification scheme. We access a kernel resource through its I.D. number in ITRON-MP. The I.D. number of a kernel resource consists of the field representing the class to which the resource belongs and the field identifying the resource in the class. When we create a new kernel resource, we must provide the I.D. number of the resource to be created. Then the hardware resource for the kernel resource can be determined from I.D. number's class field. By using this scheme, almost all system calls become compatible with those of ITRON, and porting a program developed for ITRON to ITRON-MP becomes easy.

We can express the class field of a kernel resource I.D. in two ways. The absolute method uses a class I.D. to uniquely identify a resource class in the system. The relative method specifies a class relating to the class to which the accessing task belongs. For example, the relative method is effective when a local task of a processor accesses its private resource.

In ITRON-MP, the absolute method is standard, because it agrees with typical shared-memory multiprocessor systems. We fix the I.D. number of each class upon generating the kernel. The relative method is also permissible in ITRON-MP as an option, as it is suitable for some architectures or applications. We use nonpositive values to specify a class in the relative method. In particular, we reserve zero for the code designating the class to which the accessing task belongs.

The possible values in the second field of a resource I.D. must be fixed for each class upon generating the kernel. When dynamic assignment of the kernel resource I.D. is preferable, we use a library routine that finds an unused I.D. number.

As described previously, the I.D. number of a kernel resource depends on the physical location of the resource. In an application program, the I.D. number of a kernel resource should be written using a constant symbol. Programmers have only to assign an actual value to each constant symbol when porting an application program to a new architecture. This assignment process determines the allocation of kernel resources to hardware and is very important for improving the runtime efficiency of the system.

Task-scheduling principle. Real-time task-scheduling problems for multiprocessor systems are widely studied. Many algorithms have been developed so far.[6,7] However, no task-scheduling algorithm applies to all applications. Currently, we must select an appropriate algorithm for each application. Then ITRON-MP adopts a general framework on which various task-scheduling algorithms suitable for each application can be realized.

We adopted priority-based scheduling as the base-scheduling mechanism of ITRON-MP for the following reasons:

- The task scheduling overhead is small.
- The rate monotonic algorithm, which is one of the most promising task-scheduling algorithms for hard real-time systems, implements easily on a priority-based kernel. Moreover, the priority-ceiling protocol, a task-synchronization algorithm with predictability, can also be realized on this kernel.[8]
- Dynamic time-driven, task-scheduling algorithms can also be realized as a task on a priority-based kernel by translating the scheduling table generated by the scheduling task to the priorities of tasks. If the kernel uses a dynamic task-scheduling algorithm (which has a big overhead in general), we fear that a task requiring very fast response cannot be scheduled in time. In such a situation, if the kernel works based on the priorities, we can dispatch the task immediately by skipping the dynamic scheduling algorithm.[7]
- ITRON also adopts priority-based scheduling. Compatibility with ITRON is important for ITRON-MP.

Multiprocessor systems offer many variations of priority-based scheduling algorithms. A naive way to extend priority-based scheduling for single-processor systems to multiprocessor systems involves a kernel that always minimizes the sum of the priority values of running tasks in the whole system. (A smaller priority value represents the higher priority in ITRON-MP.) Though this scheduling policy (called the naive policy) is one of the ideal policies, its implementation cost is very high and it can lead to the task-wandering problem.

One reason for the high implementation cost is the need to include an interrupt mechanism between each pair of processors. To start the execution of a task on another processor immediately, we need at least a mechanism to interrupt a processor from another one. However, because the tasks executable on multiple processors do not need to respond quickly in some systems, the requirement for this mechanism is too restrictive for ITRON-MP.

Figure 3 shows an example of the task-wandering problem. In the initial state, processor p_1 executes task t_1, which is executable on all processors; processor p_2 executes $\{p_2\}$-bound task t_2 (Figure 3a). Suppose that $\{p_1\}$-bound task t_3, which has a higher priority than t_1 and t_2, becomes ready (waiting to be executed). Since t_1 has higher priority than t_2, t_1 migrates from p_1 to p_2, and p_1 begins to execute t_3 (Figure 3b). If t_3 terminates its execution afterward, t_1 migrates back to p_1 and the system returns to the initial state (Figure 3c). When t_3 executes repeatedly within short intervals and each execution is very short (which is usually a case in some I/O processing tasks), the migration overhead of t_1 degrades the total performance of the system. In this case, it is better to make t_1 ready temporarily instead of migrating it to p_2 (Figure 3d).

The ITRON-MP specification does not determine a fixed scheduling policy or algorithm, but defines a principle that each implementation or adaptation must follow. We determine the task-scheduling principle by relieving the naive policy

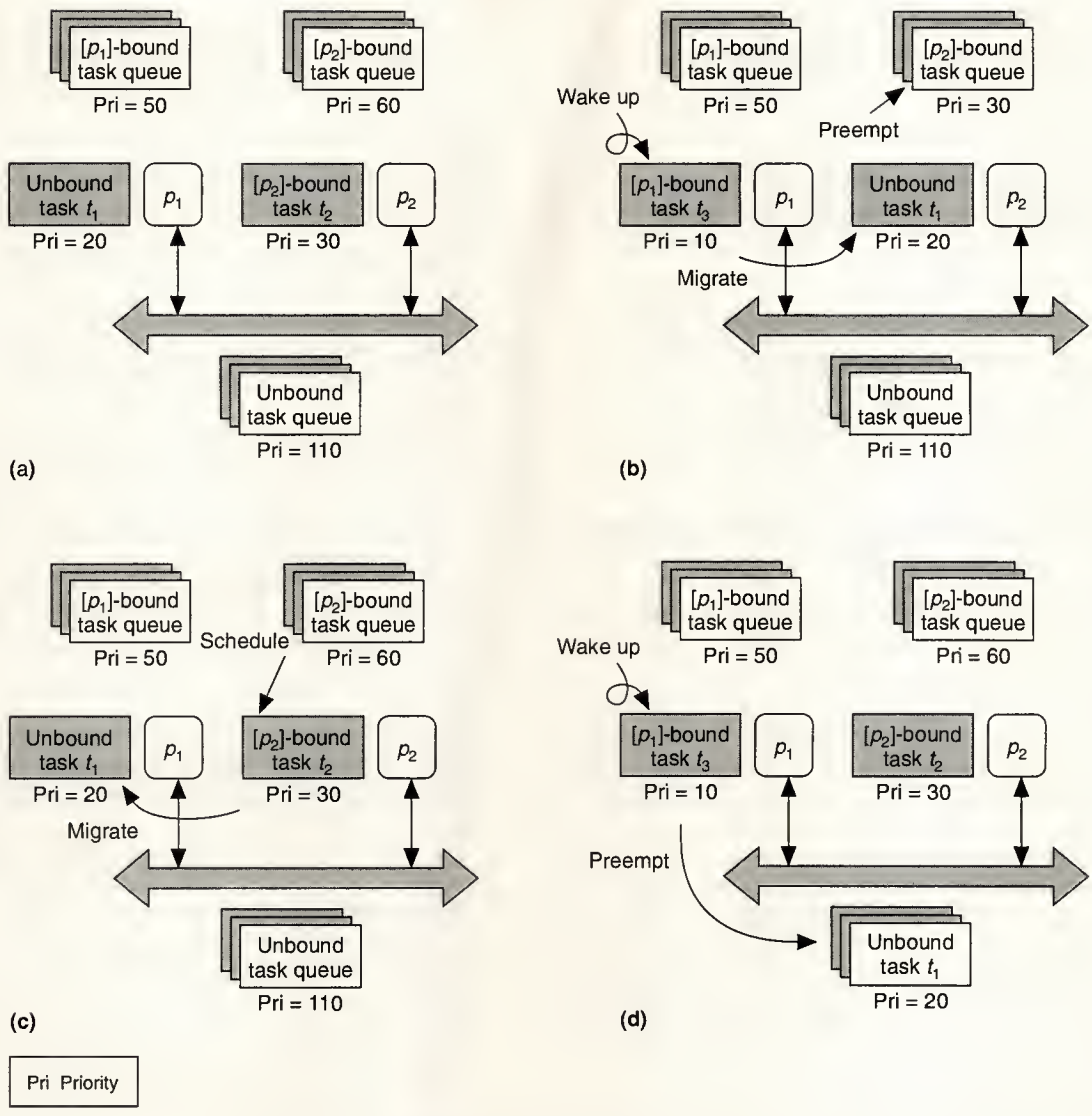


Figure 3. Task-wandering problem.

in three directions. We narrow the area in which priority values are uniformly interpreted, permit local optimal states, and allow the execution of a task deferred to some extent.

At first, priority values must be uniformly interpreted only among the tasks belonging to a class. In other words, priority values of tasks belonging to different classes may be differently interpreted. Then, a scheduling policy in which local tasks always have higher priority than global tasks is possible. Moreover, the interpretation of priority values may be different for each processor. Below, the word *priority* specifies the index for the scheduling of a task defined for each processor; priority is distinct from the word *priority value*, which is an attribute of a task.

A local optimal state occurs when each processor executes a task with higher priority than all the ready executable tasks. A local optimal state does not necessarily achieve the optimum state of the system shown in Figure 4 on the next page. We call a local optimal, but not systemwide optimum state, simply a local optimal state later in this article.

At last, the actual execution of a task may be delayed as long as another event makes task switching necessary. However, the execution of a private task must not be delayed.

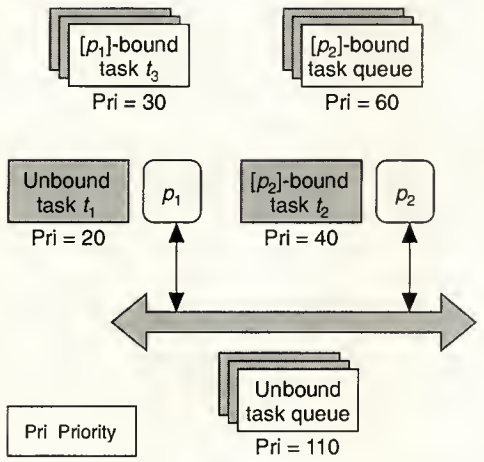


Figure 4. Local task optimal state.

A strict definition of the task scheduling principle is as follows:

1) Three task attributes determine the priority of a task for each processor: the class to which it belongs, its priority value, and the absolute time when it becomes the ready state. The priorities of tasks for a processor constitute a totally ordered set (it is possible that some tasks have the same priority).
2) Between the tasks belonging to a class, a task with smaller priority value has higher priority. If both tasks' priority values are equal, the task that becomes ready earlier has higher priority (first come, first served). When two tasks with the same priority value are preempted after executing for a while, they may be treated as having the same priority.
3) For each pair of task classes C_1 and C_2, the priority relation between the tasks for a processor meets one of the following conditions:
   a) Between a task belonging to C_1 and a task belonging to C_2, the task with smaller priority value has higher priority. If the priority values are equal, the behavior of the kernel depends on the implementation or adaptation.
   b) The priority of a task belonging to C_1 is always higher than the priority of a task belonging to C_2.
   c) The priority of a task belonging to C_2 is always higher than the priority of a task belonging to C_1.
4) When a processor switches its running task, no tasks exist that can execute on the processor and have higher priority than the task executed on the processor immediately after the switch.
5) No private tasks of a processor exist in the ready state and have higher priority than the task executed on the processor. In other words, a private task of a processor that has a higher priority than the running task must start its execution immediately after it becomes ready.

The ITRON-MP specification requests each implementation or adaptation to make the adopted scheduling policy explicit. At least, the following points must be defined:

1) *Task priority relationship between classes for each processor.* In detail, we must define one of the applied conditions in scheduling principle 3 for each pair of task classes for each processor. When applying (a), we should define how to handle tasks with the same priority value.
2) *Local optimal state.* If this state occurs, the set of processors in which no local optimal state occurs must also be defined. Scheduling principle 4 requests that the processor will switch the running task to the task with the highest priority of executable tasks. This implies that the new state is optimum in the restriction that only the processor can switch its running task. If the state is optimum in the restriction that larger set of processors can switch their running tasks, the set must be defined. For example, if the set contains all processors in a system, no local optimal state occurs in the whole system.
3) *Nonprivate task-switching delay.* Suppose that a task other than a private one should be executed but has not been executed yet. Strictly speaking, a task that is not private to a processor is in the ready state and has higher priority than one of the tasks running on the processors on which the task can be executed. Definitions of the kind of events that begin the execution of the task must be given. The following are some examples of the definition:
   - Its execution starts immediately after the case occurs.
   - When one of the processors that executes a lower priority task tries to switch its running task, the task's execution starts on the processor, except when the processor begins to execute a task with higher or equal priority.

System calls. As described previously, almost all system calls in the ITRON specification are valid for ITRON-MP in the same form. In addition, we introduce new system calls supporting multiprocessor architectures to ITRON-MP. In this section, we describe the important ones.

Some kernel resources have only one instance in the whole ITRON system, but ITRON-MP contains several instances. Ready queues and system clocks are examples of this kind of kernel resource. Because the ITRON system calls that handle this kind of resource have no parameter to specify the resource I.D., they do not meet ITRON-MP requirements in their original form. We introduced system calls to which we added one parameter for the resource I.D. The original sys-

tem calls can also be used in ITRON as the system calls that handle the default resource.

- **The mrot_rdq system call.** The rot_rdq system call rotates the ready queue on the priority level specified by its unique parameter. We can implement round-robin scheduling by calling this system call periodically. Because an ITRON implementation or adaptation generally contains ready queues, an additional parameter designating the ready queue to rotate is necessary.

  The mrot_rdq has two parameters: a task class I.D. and a priority value. Also, this system call handles some tasks with higher priority among the tasks that belong to the specified task class and that have the specified priority value as if the tasks became the ready state most recently (for example, rotates them to the end of the ready queue on that priority level). The number of tasks to be rotated depends on the implementation or adaptation. The rot_rdq system call remains in the original form and rotates the ready queue to which the issuing task belongs.
- **The mset_tim, mget_tim system calls.** When a system has some clocks, the system call to set the current value of the clock and one to read it require an additional parameter to specify which clock to set and read. The original system calls of ITRON set_tim and get_tim remain unchanged. We added the mset_tim and mget_tim system calls with one additional parameter.

Since ITRON has a quite simple task-scheduling policy, only the rot_rdq and the chg_pri system calls can change the task scheduling performed by the kernel. The task-scheduling policy of ITRON-MP is more flexible, so we introduce more system calls to effect task scheduling. A dynamic time-driven task scheduler, which is implemented as a task, uses these calls to reflect its results to the kernel.

- **The mchk_rdq system call.** The scheduling principle of ITRON-MP accepts such an implementation or adaptation that delays the execution of nonprivate tasks. For this kind of implementation or adaptation, mchk_rdq simulates a task-switching situation. By calling mchk_rdq periodically, a task to be executed begins to execute within a constant time.
- **The mbnd_tsk system call.** This call temporarily restricts the set of processors executing the specified task. It temporarily handles the task as if it belongs to another task class. We can also use the system call to restore a task to its original class. We call the task class that determines the scheduling behavior of a task the scheduling class to distinguish it from the native class. The mbnd_tsk call does not change the accessibility behavior of the specified task.

ITRON-MP separates the architecture description and the kernel description.

- **The mget_cid system call.** This call returns the scheduling class I.D. of the specified task. We can obtain the native class of a task from the class field of the task I.D.

ITRON-MP also introduces system calls to build a fault-tolerant system on an ITRON-MP-based kernel. System calls that support on-line repair by notifying the kernel of the addition or deletion of hardware modules are necessary.

- **The madd_prc, mdel_prc system calls.** In a fault-tolerant system that recovers from a processor fault by making a task executable on multiple processors, the kernel should not allocate a task to a faulty processor. After the system repairs a faulty processor and it becomes operational, the kernel must begin to allocate a task to the processor. The madd_prc system call and the mdel_prc system call notify the kernel of the repair and the fault of a processor.

  These system calls have a parameter specifying the processor I.D. that is repaired or is faulty. The parameter changes the scheduling behavior after the issuing the executable tasks on the specified processor. These system calls do not perform other treatments, including the recovery of the task being executed on a faulty processor.
- **The mdmp_tsk, mrtr_tsk, mded_tsk system calls.** A usual method of recovering tasks from faults requires a snapshot of the task status and the logs of all communication between the task and the other resources after the snapshot. To make this method possible, mdmp_tsk records various information included in the TCB, and mrtr_tsk reproduces the TCB status from the recorded information. Other task statuses, which are not managed by the kernel, can be saved and restored through normal memory access while the task is suspended. The communication logs can be recorded by using extended system calls that record the kind of the invoked system call, the parameters passed to it, and its returned value.

  These system calls not only save or restore the memory area of the TCB, they also record or reproduce the task status itself. For example, when the task waits for a semaphore, the status that its TCB is linked to the waiting queue of the semaphore is recorded or reproduced. However, these system calls do not record or reproduce the task I.D. or the page table of the task

(when the task is running in the logical addressing mode) so that the task can be reproduced on other environments.

The composite mded_tsk system call terminates the specified task immediately after executing mdmp_tsk. We use the mded_tsk for the explicit migration of a task.

- **The msnd_por, mrcv_por system calls**. When some parts of a system require much higher reliability than other parts of the system, designers often duplicate parts to increase reliability. Saving important data on two disks (mirror disk) is an example of this kind of duplication. In such a system, as a client task must communicate with two server tasks, ITRON-MP introduces new system calls that allow communication with two tasks.

  We can extend all ITRON-MP intertask communication mechanisms, except the rendezvous, to duplicate communication only with the revision of the client program. However, we cannot extend rendezvous calls without any new system calls, because the cal_por system call of ITRON executes a whole rendezvous (namely, calls a rendezvous port and successively receives a returned value). Then ITRON-MP introduces a system call named msnd_por that calls a rendezvous port and returns immediately after accepting the rendezvous by the server task and a system call named mrcv_por that waits for a reply from the server task. As these system calls may corrupt the semantics of the normal rendezvous operation, they should be used in limited situations, such as in an extended system call.

Architecture description and kernel generation

For the kernel generation of conventional operating systems, we prepared a simple language to describe the specification of the kernel to be generated. In single-processor systems, attached devices and their addresses are the only architectural differences between machines. Therefore, we usually write the architecture of single-processor systems with the specification of the kernel. However, very large architectural differences exist between multiprocessor systems and mixing the description of the architecture with the kernel specification is confusing.

Consequently, ITRON-MP separates the architecture description and the kernel description; both descriptions generate the kernel code. The architecture is described in the processor memory level. The kind and the number of hardware resources, connection structure among processors and other hardware resources, the accessibility and the access cost of hardware resources from each processor, and replaceable units are described in the architecture's description.

Figure 5 presents a description of an architecture. This example is a part of the architecture description of the TRON Box system, which was one of the testbeds of ITRON-MP.

The first definition describes the Gmico board module. The

```
module GmicroBoard(bb, sw1) = {
  processor      cpu : Gmicro200(clock=20MHz) ;
  co-processor   fpu : GmicroFPU(clock=20MHz) ;

  memory         umem[size=1M] : RAM(metric=0) ;
  memory         smem[size=512K] : ROM(metric=1) ;

  timer          tm : Timertype2(obj="gb_timer.o") ;
  sync           sr : SyncType1(src="gb_sync.c", param=(sr)) ;
  device         sio : Z8530 ;

  access         cpu  → umem ;
  access         cpu  → smem(offset=0xffc00000) ;
  access         cpu  → bb ;
  access         cpu  → tm ;
  access         cpu  → sio(offset=0xfffb0010, skip=4) ;
  access         bb   → umem(offset=0x00100000*(sw1+1)) ;
  access         bb   → sr(offset=0xfffc0000+sw1*4) ;

  intr           tm   → cpu(vector=0x42) ;
  intr           sr   → cpu(vector=0x45) ;
  intr           sio  → cpu(vector=0x44) ;
} ;

system TRONBOX = {
  bus            bb : VMEbus(metric=2) ;
  for  (i = 0 ; i<17; i++) {
                 module tbd[i] : GmicroBoard(bb, i) ;
  } ;
  module ebd : EtherIntf(bb) ;
} ;
```

Figure 5. Architecture description of TRON Box.

Processor, Coprocessor, and the Memory statements describe the kind and the number (or capacity) of a processor, a coprocessor, and memory units, respectively. The metric option in the Memory statements expresses the access cost of the memory.

An Access statement,

access A → B (offset = X, skip = Y) ;

represents that *A* can access *B* and that the address of *B* starts at *X* and is incremented by *Y*. The default value of offset is 0 and skip is 1.

An Intr statement,

intr B → A (vector = X) ;

represents that *B* can raise an interrupt on *A* and that its vector number is *X*. The Timer statement describes the timer function on the Gmicro board module. As the difference of

timer functions for each system is too large to cover with a few parameters, the system developer must write the timer-hander program and describe its name in the statement. Timertype2 specifies an external specification of the timer-handling programs. By using the Access statement and the Intr statement, we can describe the timer *tm* as accessible from the processor *cpu*, and an interrupt notifies the *cpu* of its expiration.

The Sync statement describes the mechanism to raise an interrupt to *cpu* from another module, while the Excl statement describes the mutual exclusion mechanism between processors. The Device statement defines other I/O devices, and it is not used for kernel generation.

The second definition in Figure 5 indicates that the TRON Box consists of 17 Gmicro board modules and one EtherIntf module—all of which connect to a VMEbus.

ITRON-MP IS A REAL-TIME KERNEL SPECIFICATION that is adaptable to various multiprocessor architectures and aims to become a standard real-time kernel for shared-memory multiprocessor systems. By trying to avoid excessive abstraction of kernel resources, the ITRON-MP approach permits programmers to be conscious of the execution mechanism (such as the physical location of kernel resources) and eases the development of real-time systems.

On the other hand, the location transparency of resources is important for operating systems in workstations or personal computers. In the TRON project, BTRON (the business-oriented TRON) specifies this kind of operating system. We are currently researching and developing an operating system based on the BTRON2 specification.[9] We can also use an ITRON-MP-based kernel as the basis for constructing a BTRON2 specification operating system.

Acknowledgments

We thank Toru Shimuzu, Hideo Tsubota, and Koji Hirano of Mitsubishi Electric Corporation for their useful discussions. We also thank Nobuhiko Nishio and other colleagues of the Sakamura Laboratory for their suggestions and encouragement.

References

1. V.P. Holmes and D.L. Harris, "A Designer's Perspective of the Hawk Multiprocessor Operating System Kernel," *Operating System Review*, Vol. 23, No. 3, July 1989, pp. 158-172.
2. J. A. Stankovic and K. Ramamritham, "The Design of the Spring Kernel," *Proc. Eighth Real-Time Systems Symp.*, IEEE Computer Society Press, Los Alamitos, Calif., Dec. 1987, pp. 146-157.
3. H. Monden, "Introduction to ITRON, the Industry-Oriented Operating System," *IEEE Micro*, Vol. 7, No. 2, Apr. 1987, pp. 45-52.
4. K. Sakamura, *ITRON Specification ITRON2*, TRON Association, Tokyo, 1990.
5. A.L. Hopkins, Jr., T.B. Smith III, and J.H. Lala, "FTMP—A Highly Reliable Fault-Tolerant Multiprocessor for Aircraft," *Proc. IEEE*, Vol. 66, No. 10, Oct. 1978, pp. 1221-1239.
6. W. Zhao, K. Ramamritham, and J.A. Stankovic, "Preemptive Scheduling Under Time and Resource Constraints," *IEEE Trans. Computers*, Vol. 36, No. 8, Aug. 1987, pp. 949-960.
7. N. Nishio, H. Takada, and K. Sakamura, "Dynamic Stepwise Task Scheduling Algorithm for Tightly Coupled Multiprocessor ITRON," *TRON Project 1990*, Springer-Verlag, Tokyo, Dec. 1990, pp. 43-62.
8. R. Rajkumar, "Real-Time Synchronization Protocols for Shared-Memory Multiprocessors," *Proc. Int'l Conf. Distributed Computing Systems*, IEEE CS Press, May 1990, pp. 116-123.
9. K. Sakamura, "Design Policy of the Operating System Based on the BTRON2 Specification," *TRON Project 1990*, Springer-Verlag, Dec. 1990, pp. 103-118.

Hiroaki Takada is a research associate in the Department of Information Science at the University of Tokyo. He conducts TRON project research in various areas. His interests include real-time systems, parallel and distributed computing, programming languages, and hyper-text systems.

Takada received a BS and MS in information science from the University of Tokyo. He is a member of the IEEE Computer Society, Association of Computing Machinery, Information Processing Society of Japan, and Japan Society for Software Science and Technology.

Ken Sakamura's biography and address appear on p. 15 in this issue.

Address questions concerning this article to Ken Sakamura, Department of Information Science, Faculty of Science, University of Tokyo, 3-1, Hongo 7-chome, Bunkyo-ku, Tokyo 113, Japan.

KMDS

continued from p. 35

unit. KMDS uses a similar frame structure to represent the software. See Figures 3 and 4, which illustrate the frame trees of CRT display adapters.

The rules or algorithmic procedures are invoked when the value of a slot is modified, deleted, or inserted. So, they are attached rules outside the slot, instead of rules included in the slot. Type-1 knowledge relates to the slot value assignment. In KMDS, working memory contains three global information centers: Information Banner, Need Pool, and Signal Pool. The Information Banner is the design decision exchange center between knowledge modules. The Need Pool is the frame construction information exchange center, and the Signal Pool is the frame interconnection information center. With Information Banner and other slot values, type-1 knowledge decides the value of a slot under various design conditions. For a construction slot, the produced slot value points to another frame that it needs. Moreover, type-1 knowledge evaluates the frame's specification and puts them into the Need Pool. If some important design decisions result during the evaluation, type-1 knowledge also puts them into the Information Banner to alter the other frames.

Type-2 knowledge relates to the synthesis of one construction slot. As just mentioned, for a construction slot, type-1 knowledge determines the frame that will be attached to the slot and the specification of this frame. Type-2 knowledge determines the interconnections between the attached frame and its parent frame and uses some logical nets to link signals in the signal slots of the two frames. KMDS uses three kinds of logical nets:

- net to net (an interconnection between two signals),
- net to bus (an interconnection between a signal and one signal of a signal cluster), and
- bus to bus (an interconnection between two signal clusters).

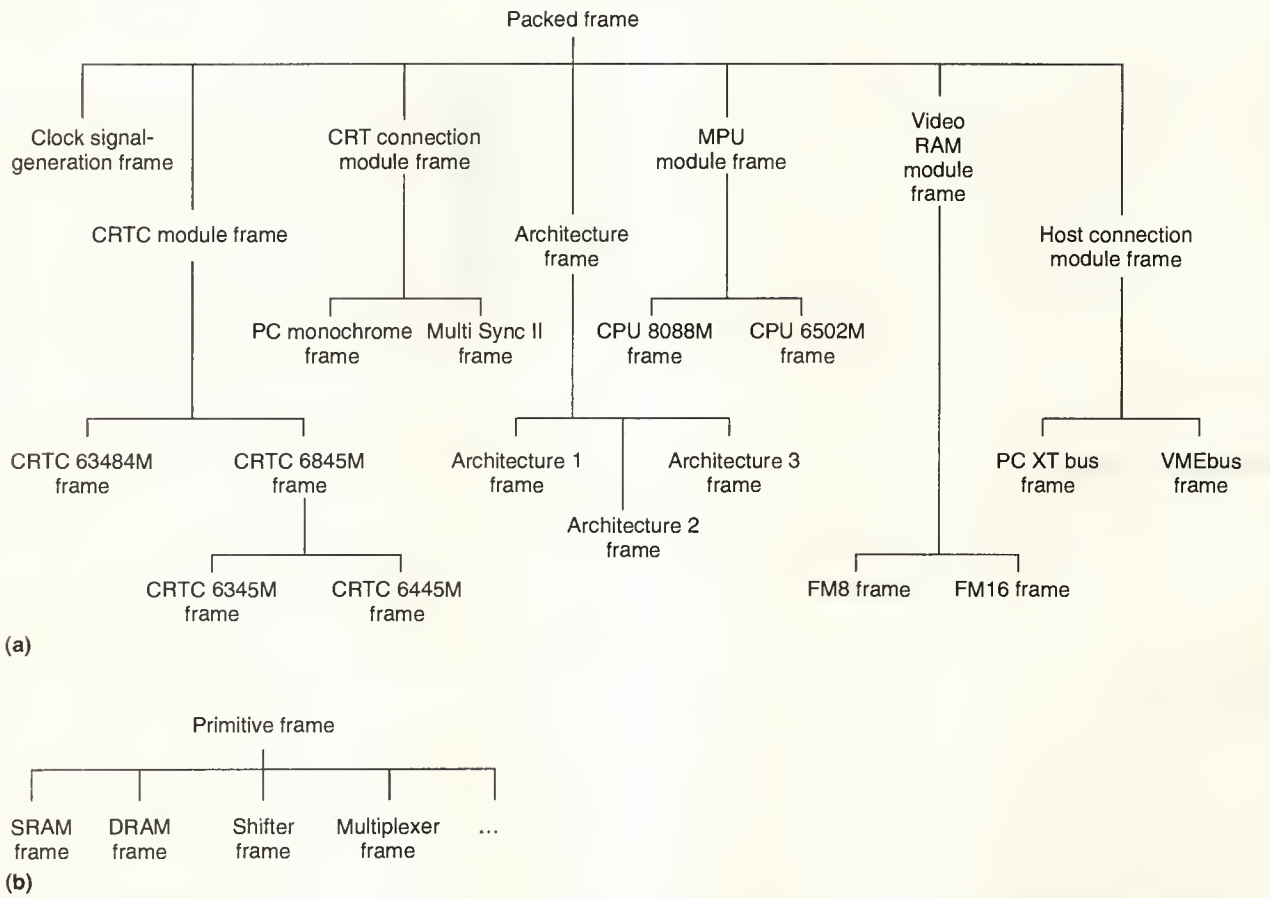


Figure 3. The hardware frame structure of CRT display adapters: packed frame tree (a) and primitive frame tree (b).

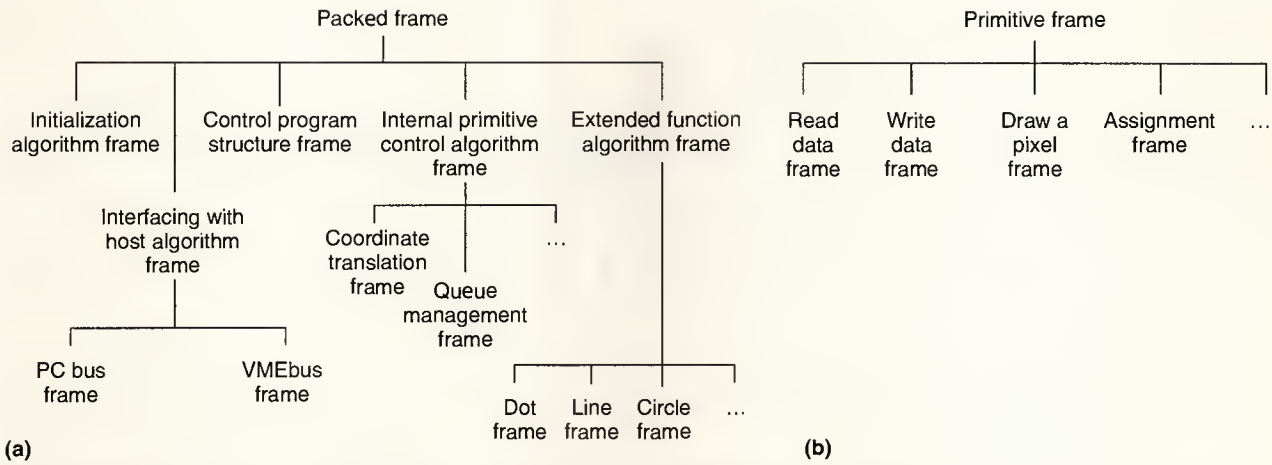


Figure 4. The software frame structure of CRT display adapters: packed frame tree (a) and primitive frame tree (b).

For software frames, a net-to-net entry indicates the procedure-call operation, but the other two logical nets are unnecessary. Sometimes, the two types of knowledge need to make a decision on multiple candidate solutions, such as architecture selection, physical device binding, and so on. In KMDS, such decision makings are based on an available factor (AF) approach in which heuristic knowledge evaluates the AF value of a candidate solution. For any decision making, the KMDS knowledge base holds two opposite groups of opinions, positive and negative. As shown in the following rule format, based on various design conditions, a positive opinion for a design strategy describes the relative availability of the strategy and expresses the degree by increasing its AF value. On the other hand, a negative opinion describes the relative unavailability of the strategy and expresses the degree by decreasing its AF value.

Rule: If Condition 1, and
Condition 2, and
...
Then the AF of candidate *i* is increased by *x*,
the AF of candidate *j* is decreased by *y*,
...
End of rule.

The adjustment of the AF value depends on expert opinions, user requirements, user design tendency, device availability, and other factors. Several adjustments may be needed in one decision-making process. KMDS picks out the one candidate with the largest final AF value as its solution. In general, the initial AF values of the candidate solutions are zeros. But, for some decisions, solutions remain heavily dependent on the four design criteria of cost, speed, functionality, and area. Therefore, KMDS evaluates a base AF for each candidate and adds it to the final AF of the candidate. KMDS evaluates the base AF on expert opinions and user tendency then implements it by the following formula:

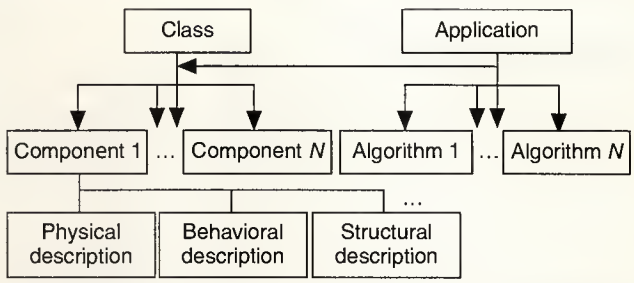


Figure 5. LSI device library organization.

$$BAF_i = (W_{Uc} * W_{Aic} + W_{Us} * W_{Ais} + W_{Uf} * W_{Aif} + W_{Ua} * W_{Aia}) * P$$

where W_{Uc}, W_{Us}, W_{Uf}, and W_{Ua} are the relative weights of user tendency on cost, speed, functionality, and area criteria, respectively; W_{Aic}, W_{Ais}, W_{Aif}, and W_{Aia} are the relative weights of the availability of solution i on cost, speed, functionality, and area criteria, respectively; P is the relative weight of the base AF on the total AF.

Both packed and primitive frames are logical components. Therefore, KMDS must bind physical devices to the constructed logical circuit. Two device libraries store physical device information: LSI and AUX IC. The libraries are also represented as hierarchical frames. Figures 5 and 6 show the organization of the two libraries. In Figure 5 the class frame and its descendent frames constitute a hierarchy of hardware devices, and the application frame and algorithm frames form

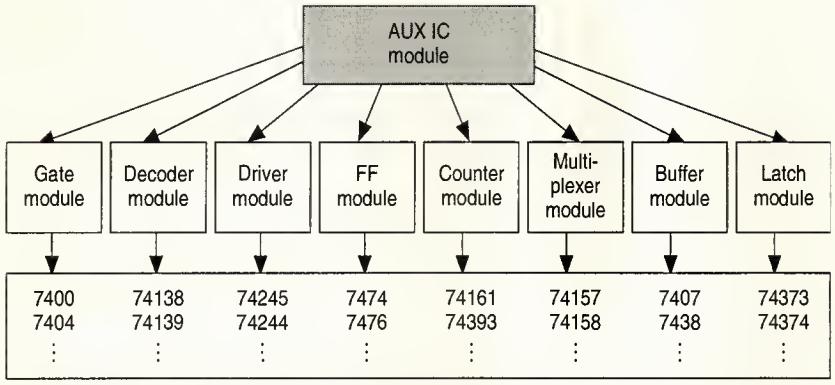


Figure 6. AUX IC library organization.

a software hierarchy. The application frame also contains a slot pointing to component frames for extracting component programming information.

Design methodology and process

KMDS uses top-down design methodology to implement a microprocessor-based digital system from a system specification description. Modularization and hierarchy directions can reduce the design space, as shown in Figure 7a and 7b. The KMDS design hierarchy follows the hierarchy of Figure 1. Since the frame structure is suitable for the methodology, KMDS introduces the frame concepts into the design process. The design process of KMDS can be formulated into a process of equation transformation:

$$\begin{aligned} MDS &= F_A(F_1, F_2, \ldots, F_n) \\ &= F_1(F_{11}, \ldots, F_{1m}) + \ldots + F_n(F_{n1}, \ldots, F_{nl}) \\ &= \ldots \\ &= P_1 + P_2 + \ldots + P_k \end{aligned}$$

MDS is the microprocessor-based digital system, and F_A is the architecture frame or control program structure frame, which is the top level of the design hierarchy. It includes several construction slots, $F_1, F_2, \ldots, F_n$, which stand for the major function units of *MDS*. From F_A, KMDS constructs each slot recursively, until the total design is constructed by primitive frames, $P_1, P_2, \ldots, P_k$.

The transformation process can be further described by the three phases shown in Figure 8: user demand analysis, hardware design, and software design.

In the first phase, a user describes the design requirements at the system level on a behavioral domain. The user interface translates them into design specifications in an internal representation format and simultaneously generates design constraints, such as cost, performance, and so on. Then the interface checks the reasonableness and completeness of design specification. Figure 9 shows the related activities and system configuration of the active user interface task. After the design specification is completed, the user demand and solution analyst constructs an architecture frame. Depending on the design specification and expert opinions in the knowledge base, the module determines all adequate architectures then selects the one with the largest available factor value as the solution. Figure 10 demonstrates this task.

In the hardware design phase, the circuit synthesizer first executes the recursive frame construction task until it achieves a set of primitive frames that can construct the selected architecture frame; that is, the logical circuit design is completed. Then the physical device binding task obtains the final circuit. A primitive frame may be bound with a physical device or multiple physical devices. Sometimes, several primitive frames may be bound with the same physical device. Finally, the synthesizer represents the circuit as a file in EDIF, for linking to external simulators or layout packages. Figure 11 on page 90 illustrates the functional blocks of the circuit synthesis task.

The software design phase generates the control program of the implemented circuit. As shown in Figure 12, this task includes control program frame construction, algorithm com-

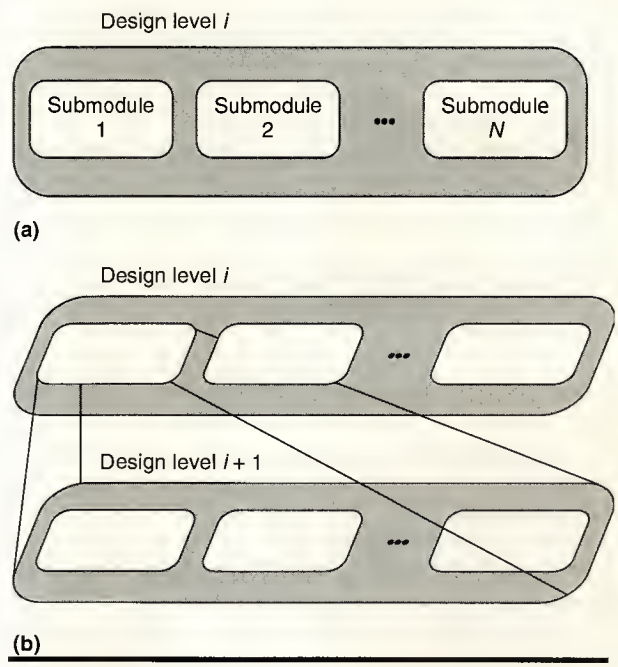


Figure 7. KMDS design methodology: modularization (a) and hierarchy (b).

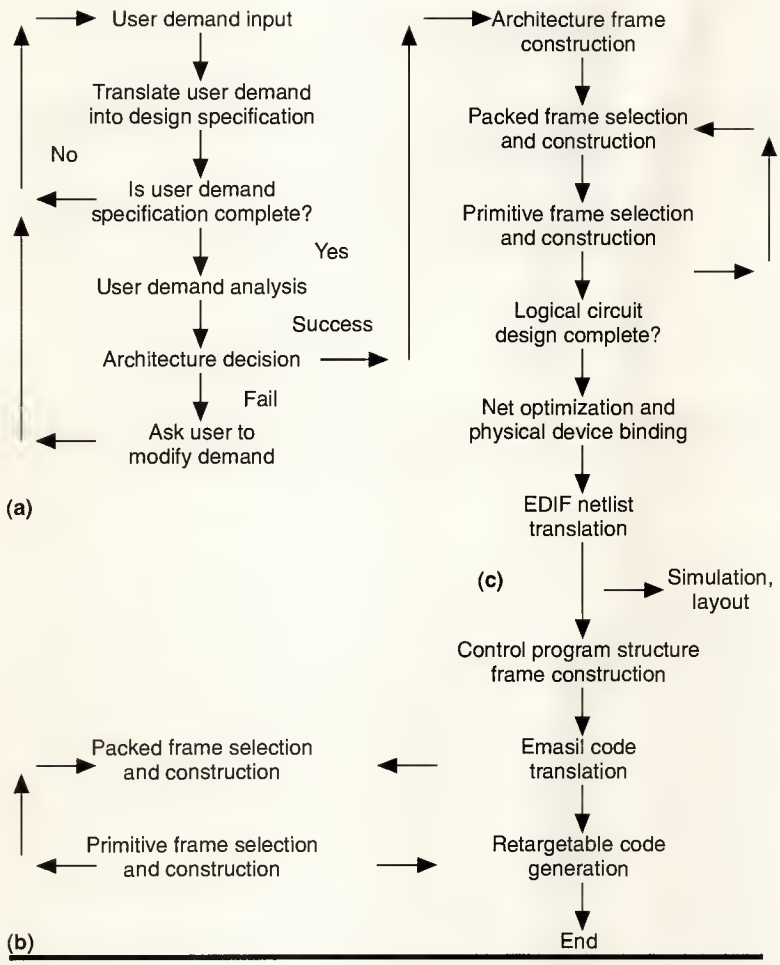


Figure 8. KMDS design process in three phases: User demand analysis (a), system software design (b), and system hardware design (c).

bination and expansion, and Emasil program generation. As hardware design, this task recursively constructs a control program structure frame until it is composed of a set of primitive frames. The primitive frames are then bound with the algorithm frames in the LSI device library.

A macro statement format represents both the primitive and the algorithm frames. Furthermore, the control program generator includes an external Emasil code generator to expand the complete control program from macro scheme to Emasil scheme, by referring the component operation mode, component programming parameters, and system design parameters. When Emasil is updated, KMDS needs only to modify this external procedure without redesigning the whole control program generator. Finally, the retargetable compiler compiles the generated Emasil program based on the instruction set of the selected microprocessor.

External utilities

Besides the knowledge-based expert system, KMDS links several external utilities to complete the whole design process. These include the Emasil retargetable compiler, NCKU-DA ASIC design system, System-Hilo logic simulator, PCDS layout package, and knowledge-base debugger. The retargetable compiler generates the machine codes of the control program of a digital system. Its high-level hardware description language is Emasil.

Emasil facilitates the description of both behavior and structure of digital systems at the system level of abstraction. Emasil is a nonprocedural language but has special constructs to specify procedural sequences; events drive nonprocedural statements, whereas sequences drive procedural statements. Nonprocedural state-

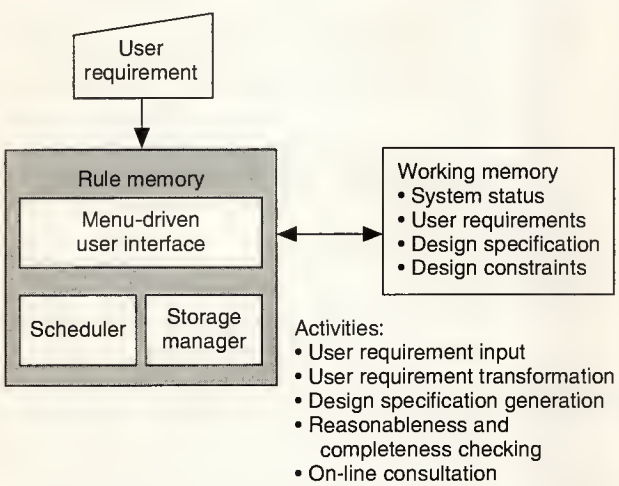


Figure 9. Interactive user interface task.

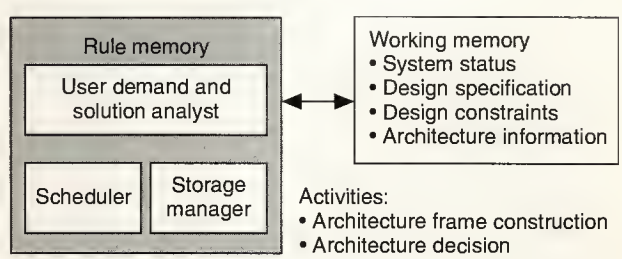


Figure 10. User demand and solution analysis task.

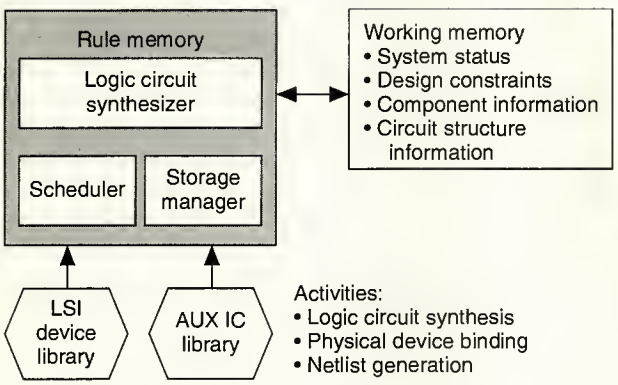


Figure 11. Circuit synthesis task.

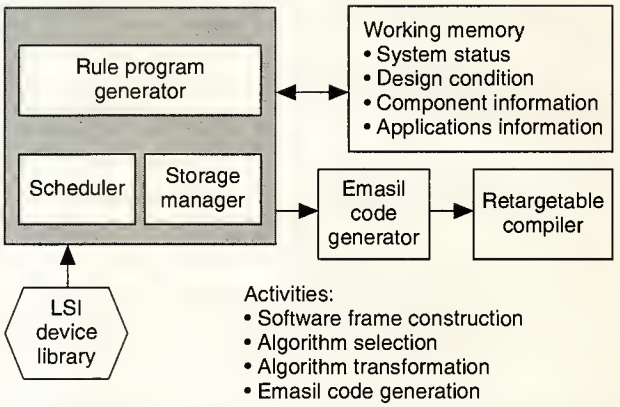


Figure 12. Control program generation task.

```
System <system_name>
    <declaration_part>
    Procedure <procedure_name>( );
        <statement_part>
    Endprocedure
        .
        . /* other procedure declarations */
        .
    Module <module_name>
        <declaration_part>
        Chip <chip_name>
            <declaration_part>
            <chip_statement_part>
        Endchip
            .
            . /* other chip declarations */
            .
        <module_statement_part>
        <interface_part>
    Endmodule
        .
        . /* other module declaration */
        .
    Processor <processor_name>
        <declaration_part>
        <statement_part>
    Endprocessor
    <system_statement_part>
    <interface_part>
Endsystem
```

Figure 13. Emasil descriptive structure.

ments specify actions associated with signals, while procedural statements describe the operations on data.

Figure 13 lists the descriptive structure of Emasil. The System declaration declares a complete digital system, such as a single-board microcomputer, including its control program. Since a microprocessor-based digital system may be composed of several subsystems, Emasil provides a Module declaration to allow designers to describe subsystems. And since a module may contain several chips, the Chip declaration describes these chips. In addition, Emasil also provides resource, label, and variable declarations.

As for the description of behavioral and structural aspects of a module or the whole system, part of Emasil contains a statement describing the former and part contains an interface specifying the latter. The statements used in Emasil include:

1) **Behavioral descriptions:** assignment, goto, inport, outport, return, case, while, for, loop, and if statements;
2) **Timing descriptions**: parallel block (PARBEG statement-list PAREND), sequential block (SEQBEG statement-list SEQEND), on, at, and wait statements; and
3) **Interconnection descriptions:** one-to-one or one-to-*N* connections may be specified.

For describing only the control program of a digital system, we need not use all the syntax of Emasil.

The organization of the retargetable compiler of Emasil appears in Figure 14. The compiler has a fixed front end and a retargetable code generator attached to a library of microprocessor description files. The retargetable code generator uses a hybrid scheme that combines the advantages of the interpretative code and table-driven code generation methods.[6,10] For each new microprocessor, designers need only create a new microprocessor description file and attach it to

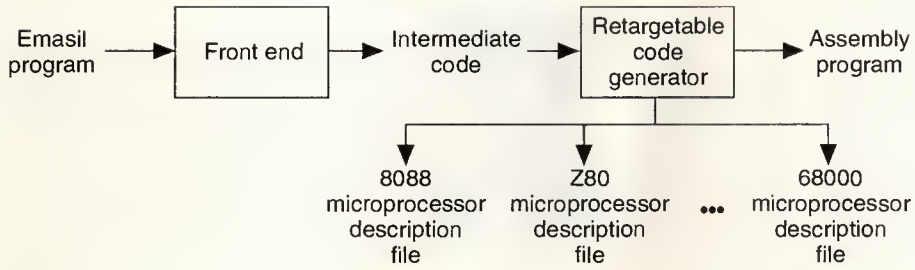


Figure 14. Organization of retargetable compiler.

the code generator. Thus, the retargetable code generator is very flexible.

As mentioned earlier, the required circuit function is sometimes too specific to find a commercial device that can realize the requirements. In this case custom chip design becomes the best choice, and KMDS invokes the NCKU-DA system. It receives a Masil program,[7] which describes the desired circuit function at the algorithmic level. Then the compiler translates the program into an internal petri net-based representation for circuit synthesis and verification.

OPS5+ and C implement the KMDS knowledge-based expert system. While OPS5+ is the tool that executes the fastest, it is not an easy tool to work with. Furthermore, the knowledge base is frequently updated. Therefore, KMDS includes a high-level knowledge-base debugger to maintain the consistency of knowledge after updating the knowledge base.

THE KMDS DESIGN SCOPE includes intelligent interface adapters and single-board microcomputers from system-level specifications. It adopts a recursive frame construction procedure to synthesize microprocessor-based digital systems by combining artificial intelligence techniques and an algorithmic approach. This approach makes KMDS flexible enough to incorporate various knowledge data into the system and to overcome the problems that result from the existence of a great deal of candidate solutions under a very high-level design specification. Another key function in KMDS is the automatic generation of a control program. This function makes the fully automatic design of digital systems possible. In fact, KMDS uses the same process in both hardware design and software design.

In addition, since KMDS attaches several external utilities, such as a retargetable compiler, ASIC design tool, knowledge-base debugger, simulator, and layout package, it constitutes a complete design automation environment. Table 2 lists the differences between KMDS and the Micon single-board computer synthesis tool developed at Carnegie Mellon University.

In efforts to compare the design efficiency of KMDS with design experts, we tried to design several CRT display adapters from various specifications generated by KMDS and human designers. We found that the costs of resultant circuits were similar and that their major difference appears in the "glue" circuits such as address decoders. The reason is that KMDS tends to use a more structured method to build the digital systems. For the selection of LSI devices, KMDS differed little from the human designers, but in designing a microprocessor-based digital system with more high-level commands, KMDS strongly tends to select high-performance, high-cost devices.

At present, we continue to pursue related research projects: 1) embedding an analogical reasoning capability into the design procedure, 2) expanding KMDS to execute testing and diagnosis, and 3) applying peehole optimization techniques to the retargetable compiler for improving the code- generation efficiency.

Readers interested in further information may contact the authors to request a 14-page KMDS design description (with schematics and program listings) of an intelligent color display adapter for an IBM PC.

Table 2. KMDS and Micon comparison.

Characteristics	Micon	KMDS
System implementation approach	Knowledge base system	Knowledge base system integrating algorithms
User specification	System	System
Knowledge representation	Rule	Frame with rule
User interface style	Query and answer	Menu driven
Design scope	Single-board computer	Single-board computer/ intelligent interface
Control program generation	No	Yes
Netlist format	Not standard	EDIF

References

1. W.P. Birmingham et al., "The Micon System for Computer Design," *IEEE Micro,* Vol. 9, No. 5., Oct., 1989, pp. 61-67.
2. N.J. Dimopoulos and H.C. Lee, "Experiments in Designing with DAME: Design Automation of Microprocessor Base Systems Using An Expert System Approach," *Proc. Int'l Computer Symp.,* 1986, pp. 1858-1867.
3. E. Horbst (eds.), *Logic Design and Simulation,* North-Holland, The Netherlands, 1986.
4. F. Hayes-Roth et al., *Building Expert Systems,* Addison-Wesley Publishing Company Inc., Reading, Mass., 1983.
5. Y.-H. Kuo, "Design Automation Assistant for Microprocessor-Based Digital Systems," Tech. Report, National Cheng Kung Univ., Taiwan, 1989.
6. Y.-H. Kuo et al., "A Retargetable Compiler for the Generation of Control Routines of Microprocessor-Based Digital Systems," *Proc. IEEE Int'l Phoenix Conf. on Computers and Communication,* 1990.
7. Y.-H. Kuo, "High-Level ASIC Design Automation Tool," Tech. Report, National Cheng Kung Univ., Taiwan, 1991.
8. H. Brown et al., "Palladio: An Exploratory Environment for Circuit Design," *Computer,* Dec. 1983, pp. 41-56.
9. Y.-H. Kuo et al., "Maintaining Consistency in a Rule-Based Knowledge System," Proc. IEEE TENCON, 1989, pp. 63-66.
10. M. Ganapathi and C.N. Fischer, "Retargetable Code Generation," *ACM Computing Surveys,* Vol. 14, No. 4, 1982, pp. 573-592.

Yau-Hwang Kuo is an associate professor in the Institute of Information Engineering at National Cheng Kung University in Taiwan. His research interests include hardware design automation, high-level synthesis, expert systems, and computer architecture.

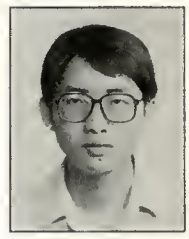
Kuo received the PhD degree in electronic engineering from National Cheng Kung University and is a member of the IEEE Computer Society.

Ling-Yang Kung is a professor in the Electrical Engineering Department at the same university. His research interests include multiprocessor systems, network computing, design automation tools, and techniques to close the hardware/software gap.

Kung received a doctorate in electronic engineering from the Politecnico di Torino in Italy. He was selected as one of the top 10 talents in informatics in Taiwan in 1990 and is a member of the IEEE Computer Society and ACM.

Ching-Chung Tzeng participated in the KMDS project while attending National Cheng Kung University. He now works as an engineer at the Communication Research Institute, Chung Li, Taiwan. His research interests include knowledge engineering and protocol engineering.

Tzeng holds an MS degree in computer engineering from National Cheng Kung University.



Guang-Huei Jeng also participated in the KMDS project before graduating from National Cheng Kung University. After serving in the military, he may accept a position as an engineer at the Communication Research Institute in Taiwan. His research interests include high-level synthesis and compilers.

Jeng received a BS degree from the Tunghai University, Taiwan and an MS degree in computer engineering from the Cheng Kung University.

Wei-Kuo Chia, another participant in the KMDS project, is now a design engineer at the Industrial Technology Research Institute's Computer & Communication Research Laboratory in Taiwan. His research interests include high-level synthesis, ASIC design, and high-performance, dedicated hardware design of computer graphics.

Chia received a BS degree from the Feng-Chia University and an MS degree in computer engineering from the National Cheng Kung University.

Address questions concerning this article to Yau-Hwang Kuo, Institute of Information Engineering, National Cheng Kung University, Tainan, Taiwan 70101, Republic of China; or via e-mail at nckut139@twnmoe10.bitnet.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 165 Medium 166 High 167

In the mailbag

(LK: liked, DLK: disliked, LTS: like to see)

June 1990

LTS: More references ...—V.C., Cochin, India (*Micro*'s Author Guide limits authors to 10 references, though most articles exceed this limit—D.D.C.)

LTS: The best application fields for RISC defined—P.B.F., Barcelona

December 1990

LK: ... detailed goals and results on parallel processing—T.S., Dekalb, IL

DLK: New format (articles continued) is terrible; LTS: addresses or phone numbers in the Product Summary—K.T.R., North Amherst, MA

DLK: Breaking articles into two parts. Make them continuous like they used to be.—B.C., Irvine, CA

LK: Parallel processing theme; too bad it was unreadable; DLK: the issue: "continued on p. *n*" considered harmful; LTS: the issue republished without "continued on p. *n*" anywhere.—R.D., Bedford, TX

DLK: That articles are split up and continued at the back. *Micro* is becoming a mess; LTS: A less messy *Micro*. Keep articles in one piece.—S.G., Newcastle Upon Tyne, England (These comments about the December redesign are the strongest disagreements we've received, and we thank you for your feedback. Since few similar comments followed, we assume readers are now accepting the design's merits.—D.D.C.)

LTS: More about RISC workstations; DLK: the cover design. I used to put it upside down when it was on my desk.—R.S., Tehran, Iran (I appreciate the fact that *Micro* is on your desk so often to cause such a problem; the back-cover advertiser will appreciate it even more. However, thanks for your frank opinion. See the June issue for the RISC chips; we will do something for systems too.—D.D.C.)

February 1991

LK: Information on optical computing and neural networks; LTS: more [of the above]—D.B., Bellevue, NE

LK: Tops! "The Drive to the Year 2000" is a good, precise review of the present and expectations for the future; LTS: USA challenges for technology leadership ... recognition of world centers of excellent R&D—D.S. Marietta, GA

LK: Futurebus articles; LTS: more high-performance bus articles—R.K., Bloomington, MN

LK: Everything ...; DLK: the idea of breaking the articles in two parts—G.M., Warsaw, Poland

LK: This issue was better than all previous ...—I.K., Osijek, Yugoslavia (Thanks for the kind appreciation. We hope you will be able to say this many times again.—D.D.C.)

LTS: [everything on] ... multimedia for and on PCs ... —W.B.K., Southfield, MI (Something will come soon.—D.D.C.)

LK: The papers on education. Nowadays ... teaching about microprocessors is really a challenging task; DLK: the papers to be broken and mixed up; please return to the old fashion. ... too detailed listing ... is redundant. Figure 1 on p. 35 shows 8085 manuals (and not 8086 ones).—J.R.C., Warsaw (Other papers on education will come; would you like to make any further proposals? You are right about the figure; thanks for the remarks and for the birthday wishes.—D.D.C.)

April 1991

LTS: More articles on disk drives, mass storage, SCSI ...—A.S., Santa Clara, CA (This is an interesting proposal; we will consider it carefully.—D.D.C.)

continued from p. 6

ing mistakes (trial and error) is part of the reverse-engineering process and has to be expected. In other technological fields, such as research and development of drugs and herbicides, profits from successful inventions pay for unsuccessful work. Drilling dry holes is something that comes with the territory, and the cost of failures has to be factored into the rate of return.

If this concept prevails on appeal, the judge's comment that maybe the jury thought AMD's reverse-engineering efforts were not the kind that counted under SPCA would be legal error, for AMD would have unquestionably shown an investment of sweat quality. Once that test was met, the substantial identity test would be triggered. If the evidence placed before the jury did not show substantial identity, as the previous summary suggests, it would seem to follow that Brooktree failed to overcome AMD's defense case.

Original mask works?

A final wrinkle adds more confusion. In addition to requiring a record of substantial toil and effort (sweat of the brow), and absence of substantial identity, the SCPA says the reverse-engineering defense is established only when the result of the reverse-engineering study is "an original mask work." Does that mean just that a chip must be more than trivially different from the original chip? Or does that mean, for example, that the second chip must contain enhancements?

The legislative history of the SCPA suggests that creation of enhancements (smaller die, higher yield, less noise sensitivity, and so on) is one of the benefits of reverse engineering. But the SCPA does not demand success in providing enhancements. Thus, a defendant that shows that its chip contains enhancements will probably win on the reverse-engineering issue. On the other hand, a defendant who failed to show any enhancements would not necessarily lose. In this case, AMD claimed that some features that it put into the Am81C458 were enhancements over Brooktree's Bt458. (I do not know whether Brooktree controverted these claims.) The record in this case does not make it possible to determine what the jury thought of the enhancement claims. I think that it is not possible, therefore, to decide the case on whether AMD proved that it enhanced its version of the chip.

If proof of enhancements is a key fact, several other inquiries must be made. First, suppose that the appellate court considers enhancements indispensable. Apparently, AMD's expert claimed that AMD had enhancements and described what they were. If Brooktree failed to controvert that testimony in any way, the jury should have been obliged to believe that AMD had enhancements. To the extent, if any, that the jury verdict must rest on a finding of no enhancement, the verdict is unsupported. If Brooktree's expert testified that the enhancements were of no value or nonexistent, however, the jury was free to believe one expert or the other. Then, the jury verdict cannot be faulted because of the enhancement issue.

Second, suppose that the appellate court considers enhancements not a necessary element of a reverse-engineering defense. Then, how the parties asked the judge to instruct the jury on this point and what he said becomes important. If he told the jury that enhancements were necessary, over AMD's objection, his instruction would be erroneous and, quite likely, justify a reversal. If neither party asked the judge to instruct the jury on enhancements, and he did not, the appellate court will not probably find a legal error that justifies reversal of the judgment. (If you don't ask for something at trial, usually you can't complain later about its absence.)

Determining the originality of a mask work may also mean more than just a rote change in the layout. Such a rote change might be a mere adaptation of the layout in the light of new design rules (such as using 0.8-micrometer line width instead of 1.2-micrometer line width, or a change in minimum radius of curvature at corners), or an arbitrary swapping of the locations of two elements. A nonrote change would certainly occur in the case of an enhancement.

Probably, the appellate court cannot deal with an issue this subtle, unless it has been properly framed in the record by the testimony of experts. That does not appear to have happened in this case, since Brooktree's counsel merely asked his expert whether he considered AMD's design "an original," and he said, "no." No doubt AMD's expert gave the same kind of conclusory "yes" answer to the same kind of question somewhere else in the trial record.

One possible way for the appellate court to avoid this issue is to observe that reverse engineering is an affirmative defense, meaning that the defendant must prove all elements of it at trial. If "original mask work" means something more than nontrivial differences, it is the defendant's responsibility to enter evidence on the issue. If the record contains no such evidence, the defendant is simply out of luck, because of its failure to prove what it was obliged to prove.

Reference

1. *Intel Corp* v. *Advanced Micro Devices, Inc.*, No. C90-20237-WAI, N.D. Cal. (complaint filed 1990).

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 177 Medium 178 High 179

1951-1991
40 YEARS OF SERVICE
IEEE COMPUTER SOCIETY

IEEE COMPUTER SOCIETY
Membership / Subscription Application

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

BENEFITS

Computer
You automatically receive *Computer* with membership. Written, reviewed, and refereed by experts, it features survey and tutorial articles covering the entire computer field, and departments such as new products, product reviews, standards, and a reader forum called "The Open Channel." (monthly).

Technical Committees
Participate in one or more of our 33 technical committees — networks of professionals with common interests in specialty areas within computer hardware, software, and applications.

Standards Working Groups
Participate in the development of the more than 100 standards projects currently sponsored by the society in such diverse areas as software engineering, local area networks, microprocessor buses, design automation, programming languages, and standards definitions.

Computer Society Press Books and Videos
Receive discounts of up to 50% on over 700 titles covering a broad spectrum of computer science topics such as networking, communications, advanced systems, image processing, security, artificial intelligence, and visualization. Over 120 new products are published annually.

Conferences and Tutorials
Choose from more than 100 conferences annually, ranging from large industry-oriented conferences replete with exhibits to small, highly interactive workshops. Members receive special low rates.

Schedule of Fees

To join: see item 1, 2, or 3.
To subscribe: see item 4.

Membership dues and periodical subscriptions are annualized to, and expire on, December 31. Pay full- or half-year rate depending on date of receipt by the Computer Society as indicated below.

	Half Year Mar 1-Aug 31	Full Year Sept 1-Feb 28
1 I don't belong to the IEEE and I want to join just the Computer Society	☐ $27.00	☐ $ 54.00
2 I don't belong to the IEEE and I want to join both the Computer Society and the IEEE*		
I reside in Region 1-6 (United States)	☐ $52.50	☐ $105.00
I reside in Region 7 (Canada)	☐ $48.50	☐ $ 97.00
I reside in Region 8 (Europe, Africa, or the Middle East)	☐ $48.00	☐ $ 96.00
I reside in Region 9 (Latin America)	☐ $44.50	☐ $ 89.00
I reside in Region 10 (Asia and Pacific)	☐ $43.50	☐ $ 87.00
3 I already belong to the IEEE and I want to join the Computer Society	☐ $11.00	☐ $ 22.00

*ACM members who join both IEEE and the Computer Society may deduct $5 off the full-year rate; $2.50 off the half-year rate.

IEEE Member Number _____

4 OPTIONAL PERIODICALS for new or current members

	issues per year		
IEEE Computer Graphics and Applications	6	☐ $12.00	☐ $ 24.00
IEEE Design and Test	4	☐ $11.00	☐ $ 22.00
IEEE Expert	6	☐ $10.00	☐ $ 20.00
IEEE Micro	6	☐ $10.50	☐ $ 21.00
IEEE Software	6	☐ $12.50	☐ $ 25.00
Transactions on:			
Computers	12	☐ $12.00	☐ $ 24.00
Knowledge and Data Engineering	4	☐ $ 7.50	☐ $ 15.00
Parallel and Distributed Systems	4	☐ $ 7.00	☐ $ 14.00
Pattern Analysis and Machine Intelligence	12	☐ $12.00	☐ $ 24.00
Software Engineering	12	☐ $11.00	☐ $ 22.00

Total amount remitted with this application $ _____

DC residents add sales tax to optional periodicals.

☐ Checks accepted in Belgian, British, German, Swiss, Japanese, or US currencies. Checks must be drawn on a bank in the country of origin of the currency.

PRICES EXPIRE 12/31/91

☐ Visa ☐ Master Card ☐ American Express

Charge Card Number (Minimum Charge $10.) | Mo. Yr. Exp. Date

I hereby make application for Computer Society membership and, if elected, will be governed by IEEE's and the society's constitutions, bylaws, and statements of policies and procedures.

Full signature _____ Date _____

☐ Male ☐ Female

MAILING ADDRESS

First name _____ Middle initial(s) _____ Last name _____ Date of birth _____

Street address _____ City _____ State/Country _____ Zip _____

EDUCATION (highest level completed) Course _____ Degrees received _____ Date _____

Name of educational institution _____

OCCUPATION

Title or Position _____

Firm name _____

Firm address _____

City _____ State/Country _____ Zip _____

ENDORSER (an IEEE member, Senior Member, or Fellow who knows you professionally)

Endorser's signature _____

Name (print in full) _____ IEEE Member No. _____

Street address _____

City _____ State/Country _____ Zip _____

Return to: IEEE Computer Society, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720–1264 USA.
Residents of Europe mail to: IEEE Computer Society, 13, Avenue de l'Aquilon, B-1200, Brussels, BELGIUM.
Asian/Pacific residents mail to: IEEE Computer Society, Ooshima Building, 2-19-1 Minami-Aoyama, Minato-ku, Tokyo 107 JAPAN.

FOR DISPLAY ADVERTISING INFORMATION, CONTACT:

Northwest Region: D. Rodney Brooks; Tel: (415) 905-0260; Fax: (415) 896-1512.

Eastern Region: Georgette Boone; Tel: (415) 905-0260; Fax: (415) 896-1512.

Southwest Region: Kenneth Smitley; Tel: (415) 905-0260; Fax: (415) 896-1512

Recruitment and Classified Advertising: D. Rodney Brooks; Tel: (415) 905-0260; Fax (415) 896-1512.

Director of Sales: Randall L. Stickrod, 544 Second St., Suite 200, San Francisco, CA 94107; Tel: (415) 905-0260; Fax: (415) 896-1512.

For production information, conference, and classified advertising, contact Heidi Rex or Marian Tibayan.

IEEE MICRO, 10662 Los Vaqueros Cir., PO Box 3014, Los Alamitos, CA 90720-1264; phone (714) 821-8380; fax (714) 821-4010.

	RS #	Page #
Allen Systems	12	46
Applied Microsystems Corp.	82	49
Aydin Controls	22	48
Bit 3 Computer Corp.	25	48
Burr-Brown Corp.	85	49
C-COR Electronics	20	47
CACI Products Co.	—	C.IV
Cal Comp	16	47
Full Source Software Corp.	27	48
Gespac	13	46
Hercules Computer Tech.	11	46
Industrial Prod. Division of IEE	17	47
Mercury Computer Systems	10	46
Multilingual Software	83	49
Notebook Computer Co.	23	48
Park Engineering Associates	80	49
Planar Systems	15	47
Rohm Corp.	14	47
Sanyo Icon	81	49
Software Interphase	84	49
Software Research	28	48
Solectek Corp.	18, 21	47, 48
Specialized Systems Consultants	26	48
Telebyte Technology	19	47
Xycom	24	48

1951-1991
40 YEARS OF SERVICE

IEEE COMPUTER SOCIETY
A member society of the
Institute of Electrical and Electronics Engineers, Inc.

THE FOLLOWING INFORMATION IS AVAILABLE:

Contact the Publications Office;
to facilitate handling, please request by number.

- Membership application, student #203, others #202
- Periodicals subscription form for individuals #200
- Periodicals subscription form for organizations #199
- Publications catalog #201
- Compmail electronic mail brochure #194
- Technical committee list/application #197
- Chapters lists, start-up procedures #193
- Student scholarship information #192
- Volunteer leaders/staff directory #196
- IEEE senior member grade application #204
  (requires ten years practice and significant performance in five of those ten)

To check membership status or report a change of address, call the IEEE toll-free number, 1-800-678-4333. Direct all other Computer Society related questions to the Publications Office.

PURPOSE

The IEEE Computer Society advances the theory and practice of computer science and engineering, promotes the exchange of technical information among 100,000 members worldwide, and provides a wide range of services to members and nonmembers.

MEMBERSHIP

Members receive the acclaimed monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others interested in the computer field.

PUBLICATIONS AND ACTIVITIES

Computer. An authoritative, easy-to-read magazine containing tutorial and in-depth articles on topics across the computer field, plus news, conferences, calendar, interviews, and product reviews.

Periodicals. The society publishes six magazines and five research transactions. Refer to membership application or request information as noted above.

Conference Proceedings, Tutorial Texts, Standard Documents. The Computer Society Press publishes more than 100 titles every year.

Standards Working Groups. Over 100 of these groups produce IEEE standards used throughout the industrial world.

Technical Committees. More than 30 TCs publish newsletters, provide interaction with peers in specialty areas, and directly influence standards, conferences, and education.

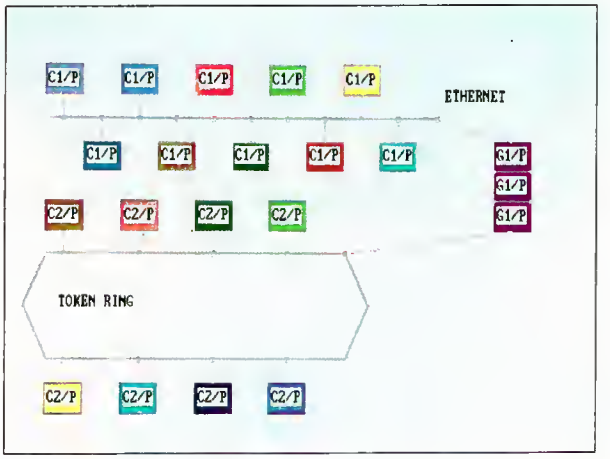
Conferences/Education. The society holds about 100 conferences each year and sponsors many educational activities, including computing science accreditation.

Chapters. Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.

OMBUDSMAN

Members experiencing problems — magazine delivery, membership status, or unresolved complaints — may write to the ombudsman at the Publications Office.

LANNET II.5
Local Area Networks



COMNET II.5
Wide Area Networks

New for local or wide area network analysts

Free trial and, if you act now, free training

LANNET II.5 uses simulation to predict your LAN performance. You simply describe your LAN and workload.

Animated simulation follows immediately --no programming.

Easy-to-understand results

You get an animated picture of your LAN. System bottlenecks and changing levels of utilization are apparent.

Your reports show LAN statistics such as transfer times, delays, and queues. Client, server, and gateway statistics show queue lengths, waiting times, and messages sent.

Your LAN simulated

You can predict the performance of any LAN. Industry standard protocols such as Ethernet, Token Ring, Token Bus, FDDI, and 10Base-T are built-in. Variations can be modeled.

COMNET II.5 uses simulation to predict your network performance. You simply describe your network, traffic load, and routing.

Animated simulation follows immediately --no programming.

Easy-to-understand results

You get an animated picture of your network. Routing choices and changing levels of network utilization are apparent.

Your reports show response times, blocking probabilities, call queueing and packet delays, network throughput, circuit group utilization, and circuit group queue statistics.

Your network simulated

You can include LAN's and multidrop lines in your model. X.25, SNA, DECnet, ISDN, SS7, fast packet, TCP/IP, token passing, and CSMA/CD are easily modeled.

Free Trial Offer

The free trial contains everything you need to try LANNET II.5™ or COMNET II.5® on your PC, Workstation, or Mainframe. Act now for free training--**no cost, no obligation.**

For immediate information

For LANNET II.5 call Eric Chapman, or for COMNET II.5 call Chris LeBaron, at (619) 457-9681, Fax (619) 457-1184. In Europe, call Nigel McNamara, in the UK, on 0276 671 671, Fax 0276 670 677. In Canada, call Peter Holt on (613) 782-2474, Fax (613) 782-2202.

University faculty should call about our special offer for research and teaching.

Rush free trial and training information for:

☐LANNET II.5 ☐COMNET II.5

Name

Organization

Address

Telephone Fax

Computer Operating System

Return to:
CACI Products Co.
3344 N. Torrey Pines Ct.
La Jolla, CA 92037
Call Eric Chapman
or Chris LeBaron
at (619) 457-9681
Fax (619) 457-1184

In Europe:
CACI Products Division
Coliseum Business Ctr.
Watchmoor Park, Riverside Way
Camberley, Surrey GU15 3YL
United Kingdom
Call Nigel McNamara
on 0276 671 671
Fax 0276 670 677

In Canada:
CACI Products Div.
200-440 Laurier Ave. W.
Ottawa, Ontario K1R 7X6
Call Peter Holt
on (613) 782-2474
Fax (613) 782-2202

Mie micro